VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Informační systém pro správu databázových systémů

IS for Database Systems Management

2015 **Bc. Martin Zwierzyna**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Zwierzyna**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Informační systém pro správu databázových systémů
IS for Database Systems Management

Zásady pro vypracování:

Katedra informatiky poskytuje výuku pěti předmětů v oblasti databázových systémů. Kromě toho je každoročně vypsáno množství bakalářských, diplomových a dizertačních prací v této oblasti. Pro tyto účely má katedra k dispozici několik serverů, na kterých jsou nainstalované databázové systémy. V akademickém roce 2012/2013 byl v rámci bakalářské práce vytvořený webový systém nazvaný DB Manager, který dokáže spravovat uživatelské účty na databázových systémech Oracle a Microsoft SQL Server. DB Manager má nedostatky, které byly objeveny až po nasazení do ostrého provozu.

Hlavní úkoly diplomanta jsou:
1. Doladění zjištěných nedostatků a jejich řádné otestovaní.
2. Rozšíření hromadného vytváření databázových účtů o další možnosti.
3. Synchronizace účtů vytvořených pomocí databázového manažeru s ručně vytvořenými účty.
4. Rozšíření systému na administraci dalších relačních SŘBD (MySQL, PostgreSQL, DB2,...).
5. Vytvoření subsystému pro zálohování/mazání/obnovování specifikovaných účtů.
6. Vytvoření online nápovědy a zhodnocení systému po nasazení do ostrého provozu.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Peter Chovanec**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

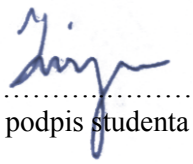
doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: *7.5.2015*



……
podpis studenta

Poděkování

Rád bych poděkoval *Ing. Petrovi Chovancovi* za odbornou pomoc a konzultaci při vytváření této diplomové práce a také *Ing. Pavlu Bednářovi* za možnost zapojení se do řešení projektu FRVŠ.

Abstrakt

Podnětem k vytvoření této práce byla potřeba vymyslet univerzální mechanismus, který by umožňoval obsluhovat a administrovat jakýkoliv klient-server databázový systém. Tím je myšleno, že zavedení nového klient-server databázového systému bude realizováno minimálním zásahem do původní implementace. Nadstavbou tohoto mechanismu bude informační systém, který bude k tomuto rozhraní poskytovat přístup. Tento systém budou obsluhovat administrátoři, kteří budou mít za úkol vytvářet a spravovat databázové účty. Systém bude ukládat informace o databázových účtech jako jejich vlastníka, důvod vytvoření, velikost účtu a jeho expiraci. Součástí systému bude také služba, která bude umožňovat provádět periodicky kontrolu databázových systémů a databázových účtů.

Klíčová slova

Databázový systém, databázové účty, administrace, synchronizace, zálohování, Microsoft SQL Server, IBM DB2, PostgreSQL, Oracle, MySQL, ASP.Net

Abstract

The main motivation behind this thesis was the need to create the universal mechanism, which would be capable of controlling every kind of client-server database system. Meaning, the new client-server database system can be implemented with minimal modifications of the original implementation. The extension for this mechanism will be information system which will be handling the access for this interface. This system is going to be operated by faculty administrators, authorized with the creation and administration of database accounts. The system is going to log the information about database accounts such as account owner, the reason for the creation, the size of an account and its expiration date. System will also contain the service for periodical checks of database systems and accounts.

Key words

Database system, User accounts, administration, automation, synchronization, backuping, Microsoft SQL Server, IBM DB2, PostgreSQL, Oracle, MySQL, ASP.NET

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
ACID	Atomicity, consistency, isolation, durability	Soubor pravidel pro transakce
API	Application programming interface	Rozhraní pro programování aplikací
BCP	Bulk copy program	Slouží ke kopírování dat do instance Microsoft SQL Serveru
CLI	Command-line interface	Rozhrání příkazového řádku
CRUD	Create, read, update, delete	Základní operace s daty
DB	Database system	Databázový systém
DBMS	Database management system	Systém řízení báze dat
DDL	Data definition language	Jazyk pro definici dat
DML	Data manipulation language	Jazyk pro manipulaci s daty
FEI		Fakulta elektrotechniky a informatiky
GUI	Graphical user interface	Grafické uživatelské rozhraní
JDB	Java debugger	Slouží pro kontrolu chyb
JDBC	Java database connectivity	Rozhraní pro přístup k relačním databázím
LDAP	Lightweight directory access protocol	Protokol pro ukládání a přístup k datům na adresářovém systému
ODBC	Open database connectivity	API pro přístup k databázovým systémům
ORM	Object relational mapping	Objektově relační mapování
SQL	Structured query language	Strukturovaný dotazovací jazyk
SQLJ	SQL java	Rozhraní mezi Java a Oracle databázemi
SŘBD	Database management system	Systém řízení báze dat
VŠB		Vysoká škola báňská
XML	Extensible markup language	Značkovací jazyk

Obsah

1 Úvod

V dnešním světě se lidé setkávají s databázemi na každém rohu, připojují se k nim a mnohdy o tom ani nevědí. Databáze jsou vytvářeny na databázových serverech. Určití administrátoři tyto databázové servery spravují a umožňují uživatelům k těmto databázím přístup. S počtem spravovaných uživatelů a spravovaných serverů náročnost aplikace rapidně stoupá. Mimo správu serverů a jejich zabezpečení plynulého chodu mají administrátoři také za úkol správu databázových účtů na běžících databázových serverech. Jedná se o náročnou práci, protože je potřeba vytvářet databázové účty s určitými právy a omezeními, upravovat je podle potřeb uživatelů a také je mazat. Z toho to důvodu je potřeba zajistit určitý přehled o těchto účtech, z jakého důvodu byl účet vytvořen, dokdy má být aktivní, či kdo je jeho vlastníkem.

Se zmiňovanými problémy při správě databázových systémů se potýkají také administrátoři na Katedře informatiky, Fakulty elektrotechniky a informatiky. Systémy spravované na této katedře slouží pro výuku mnoha předmětů, jako *Úvod do databázových systémů*, *Databázové a informační systémy I a II, Administrace databázových systémů, Java technologie* či *Vývoj informačních systémů.* Databázové předměty jsou zaměřeny na databázovou problematiku, například pro výuku základních principů fungování dotazovacího jazyka SQL, vytváření funkcí a datové analýzy vedoucí až k zavedení samotné databáze. Další skupinu předmětů reprezentují předměty týkající se vývoje informačních systémů. Jejich cílem je jednak naučit studenta vytvářet vlastní ORM, které slouží jako spojení mezi databází a grafickým rozhraním informačního systému, ale také vytvářet webové či desktopové aplikace sloužící uživateli k práci s daty v databázích. Některé z těchto předmětů jsem měl možnost absolvovat, a to bylo také hlavním důvodem k zaměření se právě na tuto problematiku. Mimo jiné se na Katedře informatiky každým rokem objeví i několik bakalářských, diplomových či dizertačních prací zaměřujících se na databázovou problematiku ne přímo využívajících databázových systémů.

Pro výuku těchto předmětů či umožnění využití databázových serverů studentům závěrečných ročníků je potřeba vytvářet velké množství účtů. Cílem této práce je vymyslet určitý mechanismus, který by dovolil administrátorům obsluhovat více druhů databázových systémů bez ohledu na jejich typ a bez předešlých znalostí administrace těchto spravovaných systémů. Úkolem je tedy navrhnout takové řešení, které by umožňovalo zavést jakýkoliv klient-server databázový systém bez větších změn v dosavadní implementaci a to bez ohledu na to, kde je cílový databázový systém umístěn.

V práci jsou jako první rozebrány základní rozdíly mezi různými typy databázových systémů a základní informace nutné k pochopení fungování jejich administrace. Dále je popsaná vize určující potřeby pro vytvoření komponenty obsluhující administrační stránku databázových systémů. Zaměřuje se na administraci pěti vybraných databázových systémů Microsoft SQL Server, Oracle Database, PostgreSQL, MySQL a IBM DB2, které momentálně běží na katedře informatiky. Další nezbytnou součástí je návrh rozhraní pro obsluhu systému a jejích příslušných komponent a částí.

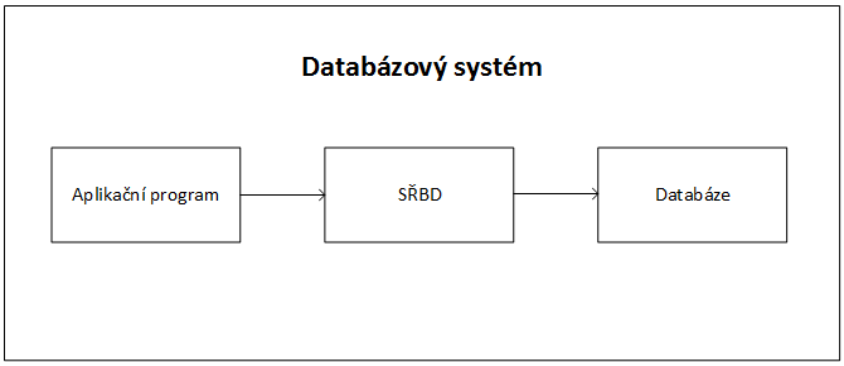
2 Základní informace o databázových systémech

V rámci této kapitoly bude popsán databázových systém, jeho částí, nástroje a architektury.

2.1 Databázový systém

Databázový systém slouží pro efektivní ukládání, modifikaci a dotazování se nad velkým množstvím dat. Skládá se ze tří částí:

- Aplikační program
- Systém řízení báze dat
- Databáze



Obrázek 2.1: *Struktura databázového systému*

2.1.1 Aplikační program

Aplikačním programem je myšlena ta část databázového systému, která umožňuje jeho správu a manipulaci s daty. Nejtriviálnějším příkladem takového aplikačního programu je nadstavba příkazového řádku jednotlivých databázových systémů.

2.1.2 Systém řízení báze dat

Systém řízení báze dat (SŘBD), neboli anglicky Database Management Systém (DBMS), je označení softwaru, sloužícího jako mezivrstva mezi uloženými daty a aplikacemi. Primární funkcí SŘBD je pracovat s daty, ukládat je, upravovat, mazat či nad nimi provádět příkazy. Nelze jej označovat za databázový systém, protože databázový systém se ve skutečnosti skládá z aplikačního programu, databáze a systému řízení báze dat. Hlavním nástrojem je strukturovaný jazyk SQL.

Hlavní úkoly SŘBD jsou:

- tvorba nových databází a jejich schémat,
- provádění dotazů nad daty, jejich získávání a úprava,
- ukládání velkého množství dat a poskytování efektivního přístupu,
- zajištění trvanlivosti a odolnosti vůči výpadkům a chybám,
- poskytovat přístup více uživatelům současně.

2.1.3 Databáze

Databázi lze nazvat určitou uspořádanou množinu informací či dat. Data jsou uložena na paměťovém médiu. Jsou uložena v tabulkách, které mají mezi sebou určité vazby. Vazby jsou reprezentovány za pomocí tzv. primárních a cizích klíčů. Každá tabulka obsahuje atributy, které představují její sloupce, nebo také záznamy, které představují jednotlivé řádky. Dále se v databázích objevují objekty jako indexy, procesy, pohledy atd.

2.2 Jazyk SQL

Jedná se o strukturovaný dotazovací jazyk určený pro práci s daty relačních databází. Je součástí systému řízení báze dat. Příkazy jazyku SQL se dělí na 5 základních skupin:

- **Data query language** - příkazy pro dotazování se nad daty (Select)
- **Data manipulation language** - příkazy pro manipulaci s daty (Insert, Update, Delete)
- **Data definition language** - příkazy pro definici dat (Create, Alter, Drop)
- **Data control language** - příkazy pro správu práv (Grant, Revoke)
- **Data transaction language** - příkazy pro správu transakcí (Start transaction, Commit, Rollback)

2.3 ACID

Jedná se o soubor pravidel [21], který zajišťuje, že jsou databázové transakce zpracovány spolehlivě. V souvislosti s databázemi se transakcí označuje jakákoliv operace nad daty. Nejjednodušším příkladem transakce je problém převodu finančních prostředků z jednoho účtu na jiný. Celý tento proces je označován za transakci, protože připsání částky na cílový účet může byt provedeno pouze, pokud byla částka ze zdrojového účtu odepsána a naopak.

- **A** – **Atomičnost** (Atomicity) – operace je dále nedělitelná, provede se vše nebo nic,
- **C** – **Konzistence** (Consistency) – při a po provedení transakce není porušeno žádné integritní omezení,
- **I** – **Izolovanost** (Isolation) – operace prováděné uvnitř transakce jsou skryty před vnějšími operacemi,
- **D** – **Trvanlivost** (Durability) - změny po úspěšné transakci jsou trvale uloženy v databázi.

2.4 Systémový katalog

Jedná se o nesdílnou součást každého databázového systému. Systémový katalog umožňuje získávat systémové údaje o daném databázovém serveru pomocí standardního dotazování za použití SQL příkazů. To znamená, že systémové údaje lze zjistit jak ze systémových nástrojů jednotlivých databázových systémů, tak i za pomocí dotazování se například z prostředí webové aplikace (za pomocí připojení prostřednictvím k tomu určeného ovladače). Příkladem může být získání seznamu všech uživatelských účtů na daném systému, získaní seznamu všech tabulek daného uživatele, atd.

2.5 Architektura databázových systémů

2.5.1 Klient-Server databázové systémy

Tento typ systému funguje na principu Klient-Server, jedná se o dvojici typů a to klient, který představuje koncového uživatele a typ server, který představuje centrální bod, ke kterému se klient připojuje. Pro zavedení se vyžaduje instalace databázového serveru na operační systém. Je určený tam, kde chceme poskytovat globální přístup k databázím pro dané uživatele. Klient-server databázové systémy mají centrální uložiště dat, o které se stará serverová část a k ní se můžou připojovat stovky klientů. Pokud vyžadujeme jednouživatelský přístup, tak pro nás nemá tato architektura smysl.

Mezi nejznámější Klient-Server databázové systémy patří:

- Microsoft SQL Server [16][17]
- Oracle Database [14]
- PostgreSQL [19]
- MySQL[18]
- IBM DB2 [15]
- Firebird [22]
- Sybase [23]

Výhody

- **Plná podpora SQL** – kromě CRUD příkazů, systémy podporují odlaďovací i administrační příkazy
- **Přístupová práva** – systémy poskytují víceuživatelský přístup a s tím související nastavení práv
- **Nástroje pro optimalizaci** – velkou výhodou je možnost ladění databáze pro specifické využití
- **Klient-server** – veškeré operace jsou prováděny na serveru, klient pouze posílá příkazy

Nevýhody

- **Složitější instalace** – z důvodu nutnosti nastavení prvotních práv, umístění databáze a dalších parametrů nutných pro zavedení databázového systému je instalace poněkud složitější
- **Nároky na HW** – vzhledem k víceuživatelskému přístupu jsou zde kladeny větší nároky na HW
- **Horší výkon** – je zde určitý pokles výkonu, který je způsoben komunikací po síti

2.5.2 Embedded databázový systémy

Embedded databázový systém je úzce svázaný s aplikací a nevyžaduje instalaci. Aplikace má přímý přístup k uloženým datům pomocí knihoven daného databázového systému, což se projevuje i na výkonu. U mnoho aplikací, které si ukládají uživatelské údaje či nastavení a nechtějí pro to používat síťového připojení (čili ukládat si data pouze lokálně), je využito právě embedded databází (webové prohlížeče, hry, mobilní aplikace a jiné.). Zavedení se realizuje pouze připojením knihovny například k aplikaci, která slouží pro obsluhování samotné databáze.

Mezi nejznámější embedded databázové systémy patří

- SQLite [24]
- Microsoft Access [25]
- Berkeley DB [26]
- MySQL Embedded [27]

Výhody

- **Jednoduchost** – poskytují to, co aplikace potřebuje, čili základní SQL funkce nutné k obsluze dat
- **Nízké hardwarové nároky** – jelikož zde odpadá nutnost instalace, připojení knihovny nezabere mnoho místa na disku či místa v paměti
- **Snadná instalace** – instalace se skládá z pouhého nalinkování knihovny a vytvoření databázového souboru
- **Odladěnost** – vzhledem k omezené funkčnosti, není potřeba řešit ladění příkazů
- **Rychlost** – absence komunikace po síti a přímý přístup k datům dělá tyto databáze daleko rychlejší

Nevýhody

- **Přístupová práva** – k databázovému souboru může přistupovat najednou pouze jeden uživatel, není zde žádný přístupový systém a nastavení práv
- **Zamykání** – pouze jedna aplikace může zapisovat najednou, zamyká se pak celá databáze, nikoliv pouze tabulka
- **Omezená podpora SQL** – podpora je minimální, řeší pouze základy SQL a nejsou zde žádné administrační příkazy

Vzhledem k tomu, že tato práce se věnuje správě klient-server databázových systémů, budu se v dalších kapitolách věnovat pouze těmto systémům.

3 Správa databázových systémů

V rámci této kapitoly bude popsána základní vize správy klient-server databázových systémů a její implementace. Před samotnou vizí je popsáno několik základních databázových pojmů nezbytných pro správné pochopení problematiky.

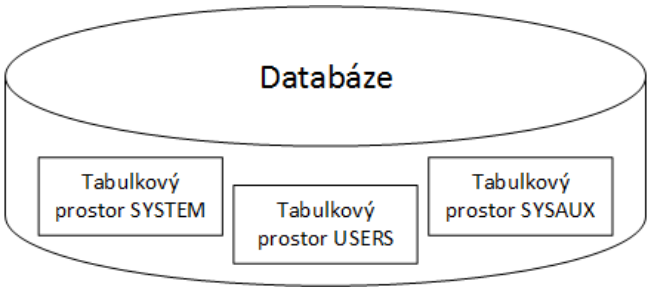
3.1 Základní pojmy

Databáze – databáze je kolekce dat uložených na disku. Může být uložena v jednom nebo i více souborech umístěných na databázovém serveru. Samotná databáze se skládá z podprostorů zvaných tablespace.

Tablespace – neboli tabulkový prostor, slouží pro oddělení logické struktury od úložných struktur

Schéma – jedná se o metainformace popisující data v databázi.

Kvóta – jedná se o číselné omezení velikosti diskového prostoru pro daného uživatele. Kvóta má za úkol řešit situaci, kdy by se uživatel snažil zaplnit databázi tak velkým objemem dat, který mu nenáleží. Kvótu lze nastavit jak u databáze, tak i u konkrétního tabulkového prostoru.



Obrázek 3.1: *Vztah databáze a tabulkového prostoru*

Uživatelský účet – je konto, prostřednictvím kterého se uživatel připojuje k databázi. Je dán svým názvem a heslem. Databázový systém může také rozlišovat i více typů uživatelských účtů, což si ukážeme až v další kapitole.

Právo – jelikož databázové systémy jsou velmi pokročilé a nabízejí obrovské množství funkcí, je potřeba uživatelům zamezit přístup k jejím částem či databázím samotným. Toho lze dosáhnout nastavením práv pro daný uživatelský účet. Právě u studentů se očekává pouze základní přístup k databázi zahrnující vytváření tabulek, zápis a čtení do nich, případně využívání procedurální nadstavby (pokud to server umožňuje). Neměli by třeba mít přístup k systémovým katalogům, které v sobě udržují informace o celém databázovém systému.

Role – role je typ uživatele s předdefinovanými právy. Pokud chci používat stejná práva pro více uživatelů, vyplatí se spíše vytvořit roli a daná práva k ní přiřadit. Poté stačí místo přidělování všech práv k uživatelskému účtu přidělit pouze roli, která mu práva sama přidělí. Role řeší i situaci, kdy bychom chtěli skupině uživatelských účtů změnit práva, stačí nám v tomto případě upravit práva pouze u role, není potřeba upravovat práva každému účtu zvlášť.

3.2 Vize

Základní vizí tohoto projektu je vytvořit interface, který bude obsahovat sadu funkcí pro správu databázových účtů na jakémkoliv klient-server databázovém systému. Funkce, které potřebujeme nad databázovými systémy provádět, jsou:

- **připojení, odpojení** – nutná podmínka pro jakoukoliv práci s databázovým systémem,
- **vytvoření databázového účtu –** vytvoření prostoru na databázovém systému zahrnující uživatelský účet, databázi nebo tabulkový prostor.

  Vytvoření účtu se skládá z následujících operací:
  - vytvoření uživatelského účtu pro zajištění přístupu,
  - vytvoření databáze, či tabulkového prostoru,
  - udělení práv pro přístup k databázi či tabulkovému prostoru,

- **zrušení databázového účtu** – smazání uživatelského účtu a databáze,
- **odblokování účtu a změna hesla**
- **změna parametrů databázového účtu –** změna parametrů jako například kvóta,
- **záloha a obnovení** – provedení zálohy databáze a její případné obnovení (záloha a obnova dat je blíže popsána v kapitole 3.3).

S každým účtem je nutné uchovávat následující informace:

1. název účtu
2. vlastník účtu
3. kdo účet vytvořil
4. kvóta
5. databázový systém, ve kterém je účet vytvořený
6. datum konce platnosti
7. zda je potřeba učet zálohovat
8. zda má být účet po skončení platnosti smazán

3.3 Záloha a obnova dat

3.3.1 Záloha

Záloha je nezbytnou součástí každé databáze, kdy chceme předejít ztrátě dat v případě selhání datového či kontrolního souboru [21]. Rozlišujme dva druhy zálohy, podle toho v jakém stavu se nachází databáze:

- **Offline** (konzistentní) – databáze je zavřená a všechna data z cache jsou uložena v databázi, žádný uživatel se nemůže k databázi připojit
- **Online** (nekonzistentní) – záloha probíhá za plného provozu databáze

Zálohují se především datové a kontrolní soubory. Záloha dalších souborů pak může být záležitostí jednotlivých SŘBD. U datových souborů nemusíme vždy zálohovat celý datový soubor.

Obvykle je možné zálohovat jen některé tabulkové prostory. Další běžné podporované typy zálohy jsou:

- **Plná záloha** – jedná se o kompletní zálohu, která může být použita k zotavení databáze
- **Inkrementální, rozdílová –** záloha, při které ukládáme pouze bloky souborů změněné od poslední zálohy, inkrementální záloha potřebuje předchozí zálohy, aby byla kompletní

Pro zálohování v případě této práce byl vybrán typ plné zálohy.

3.3.2 Obnovení

Obnovení může mít v databázi mnoho podob. Obecně se jedná o uvedení databáze do konzistentního stavu v případě nějaké chyby. Chyby jednotlivých SQL příkazů či procedur jsou obvykle řešeny v rámci běhu databáze. Rozlišujeme tyto typy obnovení:

- **Obnovení instance** – nastává po vyskytnutí systémové chyby, obnovení se provádí od posledního kontrolního bodu
- **Obnovení databáze –** nastává v případě chyby média, obnovení se provádí od poslední zálohy datového souboru

Tato funkce nebyla v konečném důsledku žádoucí, proto nebude v práci dále rozváděna.

3.4 Administrace - rozhraní pro správu

V rámci systému bylo navrhnuto univerzální rozhraní IConnections, které obsahuje následující metody:

- bool Connect() – připojení k databázi
- bool CreateLogin(String owner, String dbname, String password, int quota) – vytvoření databázového účtu za pomocí volání procedur
- bool DeleteLogin(String owner, String dbname) – smazání databázového účtu za pomocí volání procedur
- bool SetQuota(String dbname, int quota) – změna kvóty za pomocí volání procedur
- bool Backup(String owner, String dbname) – provedení zálohy vybrané databáze
- int GetQuota(String dbname) – získání kvóty pro daný účet ze systémového katalogu
- List<String> GetLogins() – získání všech databázových účtů na vybraném databázovém systému
- bool PasswordReset(String owner, String password) – restartování hesla databázového účtu
- bool CreateProcedure(String sql) – vytvoření procedur nutných pro správnou funkčnost
- bool CheckProcedures() – kontrola existence procedur
- bool Close() – ukončení připojení

Každý záznam databázového systému má daný svůj typ. (Příklad: Přidávám připojení k databázovému systému SQL Server, jeho typ bude MSSQL.). Jelikož nelze předem určit, o jaký typ databáze se jedná a jakou instanci které třídy vytvořit. Bylo potřeba vymyslet mechanismus, který bude vytvářet instance pro databázové systémy dynamicky, na základě jejích typu získaného ze záznamu v SQLite databázi. Proto je nesmírně důležité zachovat stejné názvy tříd, korespondující s typy databázových systémů.

Aplikace rozhraní

```
IConnections conn = (IConnections)Activator.CreateInstance(Type.GetType("Connections."
+ type + ",Connections"), connectionString);
```

Proměnné *type* a *connectionString* jsou typu String. Type představuje typ databázového systému a connectionString představuje řetězec definující připojovací údaje daného databázového systému. Následná obsluha těchto databázových systému je vzhledem ke stejnému rozhraní stejná a není třeba dále rozlišovat typ systému.

4 Spravované databázové systémy

Tato kapitola bude věnována pěti vybraným databázovým systémům, které aktuálně běží na katedře informatiky a vyžaduje se jejich administrace.

- Microsoft SQL Server
- Oracle
- PostgreSQL
- MySQL
- IBM DB2

4.1 Microsoft SQL Server

Microsoft SQL Server [3][4] je relační databázový systém vyvinutý společností Microsoft. Jakožto databázový systém se jedná o softwarový produkt, který má jako primární funkci ukládání a načítání dat podle požadavků specifikovaných uživatelem nebo aplikací. Jeho primární dotazovacími jazyky jsou T-SQL a ANSI SQL.

Obrázek 4.1: *Logo SQL Serveru*

4.1.1 Edice

Microsoft SQL Server [16] nabízí různé edice určené pro vybraný typ koncových uživatelů. V dnešní době jsou na trhu následující edice:

Express Edition

„Light“ verze SQL Serveru, určena pro vývojáře aplikací. K těmto účelům je zde Express Manager (XM), podpora Common Language Runtime a nativní XML. K dispozici je zde i SQL Server Management Express sloužící ke správě databáze poskytující přehled nad vyvíjenými projekty. Vše je k dispozici zdarma ke stažení na stránkách MSDN Microsoft.

Workgroup Edition

Edice navržena pro malé podnikatelské společnosti a pro použití na úrovni pracoviště. Tato edice poskytuje podporu relační databáze. Podporuje 2 procesory a maximálně 2 GB paměti.

Standard Edition

Navržena pro malé a středně velké společnosti. Podporuje až 4 procesory a také až 2 TB paměti. Dále zahrnuje celou škálu funkcionality Bussines Inteligence, analýzu, přehled a integrace služeb.

Web Edition

Web Edition je určen pro webhostingové poskytovatele. Kromě databázového rozhraní tato edice přináší přehled služeb. Podporuje až 4 procesory a 2 TB paměti.

Enterprise Edition

Jedná se o speciální formu SQL Serveru, která je určena pro časově kritické aplikace s velkým počtem uživatelů. Na rozdíl od Standard Edition, tato edice obsahuje další funkce, které mohou být užitečné pro high-end instalace se symetrickými multiprocesory nebo clustery. Nejvíce důležitou funkcí Enterprise Edition je rozdělení dat a online údržba databáze.

Developer Edition

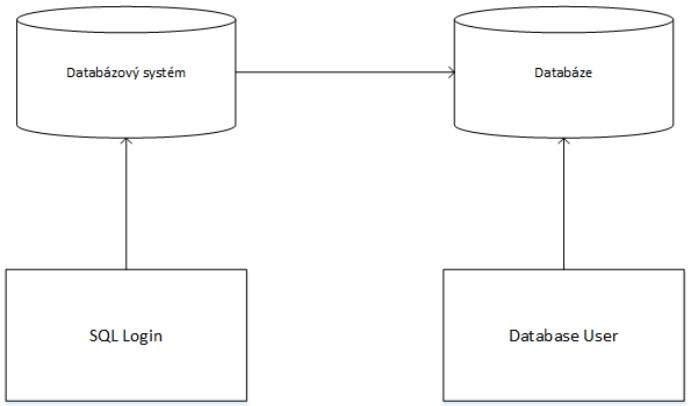
Umožňuje uživatelům vytvářet a testovat jakýkoliv typ aplikace na SQL Serveru. Obsahuje veškerou funkcionalitu z Enterprise Edition, ale je licencován pouze k použití při vyvíjení, testování či demonstrování softwaru. Pro rychlé nasazení do výroby, databázový systém Developer Edition lze velmi snadno upgradovat na Enterprise Edition.

Datacenter Edition

Nové vydání SQL Serveru, které je navrženo pro podporu nejvyšší úrovně škálovatelnosti. Nemá žádné omezení paměti, uživatel může vytvořit až 25 jeho instancí. Dále také podporuje maximálně až 265 logických procesorů.

4.1.2 Administrace

Microsoft SQL Server dokáže pracovat ve dvou režimech autentifikace [17], první se nazývá Integrated Windows Authentication a představuje přihlašování za pomocí klasického účtu systému Windows. Druhým způsobem je více oblíbený SQL Server Authentication, který pracuje s vlastní databází uživatelů, tudíž není v tomhle ohledu závislý na operačním systému Windows. Dalším způsobem rozdělení je na SQL Login a Database User. SQL Login představuje uživatelův přístup k celému databázovému systému, kdežto Database User je již přiřazení ke konkrétní databázi (Obrázek 4.2).



Obrázek 4.2: *Srovnání SQL Login a Database User*

Dalším prvkem je databáze. Každému uživateli je potřeba vytvořit databázi, která bude přístupná jenom jemu a nikomu jinému. Toho lze docílit tím, že nastavíme vybraného uživatele jako vlastníka databáze a nikomu jinému k ní nepřidělíme přístupové právo. Při vytváření se také nastaví kvóta databáze, která se ovšem může změnit (pouze zvýšit).

Připojení

SQL Server poskytuje velké množství řešení pro různá vývojová prostředí. Záměrem je nesoustředit se pouze na uživatele vyvíjející v jazycích .NET, k čemuž má SQL Server vzhledem ke společnosti Microsoft nejblíže, ale i na jiné jazyky jako C/C++, Java nebo Python.

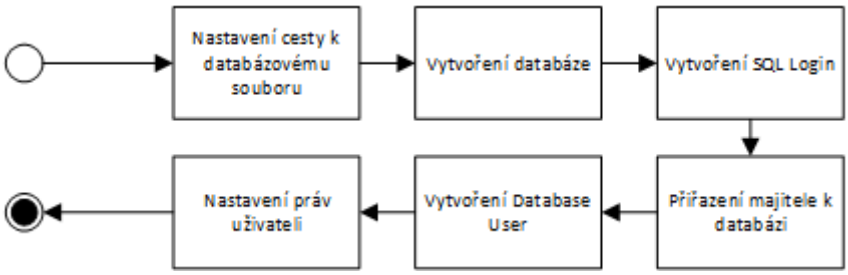
V rámci této aplikace je pro připojení využito ADO.NET, což je jádro určené pro konektivitu s databázemi prostřednictvím jazyků .NET. Zavádí se pomocí „System.Data.SqlClient“ pro přístup k SQL Serveru.

Vytvoření účtu

Vytváření databázového účtu začíná určením adresy souboru pro uložení databáze. Rozlišujeme dva případy, první kdy se databáze ukládá do implicitní cesty nastavené v databázovém systému, nebo do adresy danou uživatelem. Dále se vytvoří databáze na získané adrese a určí se její kvóta. Vytvoří se SQL Login s vygenerovaným heslem a přiřadí se mu databáze. Teď když již máme vytvořeného SQL Login uživatele a databázi, můžeme vytvořit Database User pro danou databází. Zbývá nastavit práva a také přiřadit uživateli roli BULKADMIN, která zajistí možnost zápisu velkého objemu dat. Výsledkem je vytvořená databáze s jejím přiděleným uživatelem. Implementace vytváření účtů je k nahlédnutí v příloze A.

Nastavována práva

- CREATE TABLE – právo vytvářet tabulky
- ALTER – právo umožňující práci s objekty (obnáší příkazy alter, create a drop)
- CONTROL – právo umožňující obsluhu vlastní databáze

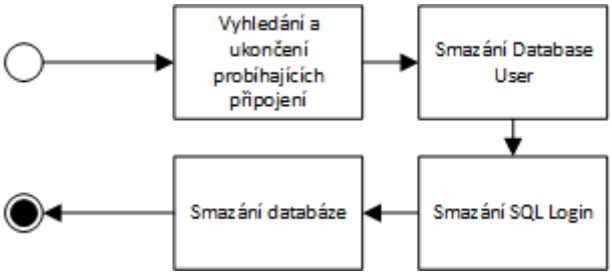


Obrázek 4.3: *Microsoft SQL Server – vytvoření databázového účtu*

Zrušení účtu

V případě mazaní databázových účtů je potřeba smazat všechny předtím vytvořené položky jako SQL Login, Database User a samotnou databázi. Za dobu testování jsem ovšem přišel na to, že je potřeba ošetřit ještě jednu věc. V případě, že uživatel není aktuálně připojen ke své databázi, by

proběhlo mazání bez chyby. Pokud by ovšem uživatel připojený byl, došlo by k chybě, která by nedovolovala mazání z důvodu existujícího připojení daného uživatele, kterého se snažíme smazat. Pro ošetření jsem použil dotaz na systémový katalog, který mi zjistí aktuální připojení tohoto uživatele. Tyto připojení následně ukončím zavoláním příkazu KILL + číslo procesu připojení. Až poté lze s jistotou provést úspěšné smazání všech potřebných položek. To musí mít ale danou posloupnost, musí se smazat Database User, poté SQL Login a jako poslední databáze. Pokud by došlo k volání příkazů ve špatném pořadí, mohlo by dojít k chybám upozorňujícím na mazání databáze i když existuje její vlastník, apod. Implementace mazání účtů je k nahlédnutí v příloze B.



Obrázek 4.4: *Microsoft SQL Server – smazání databázového účtu*

Restart hesla

Jedná se o jednoduchý příkaz, kdy je změněno přístupové heslo danému SQL Loginu a provedeno obnovení přístupu k databázi. Obnovení se provádí z důvodu, kdy po několika neúspěšných přihlášeních dochází k uzamčení účtu. Samotná změna hesla by v tomto případě neměla smysl.

Změna hesla:

```
"ALTER LOGIN [" + login + "] WITH PASSWORD=N'" + pass + "'"
```

Odemknutí účtu

```
"ALTER LOGIN [" + login + "] ENABLE"
```

Změna parametrů účtu - změna kvóty

Jedná se o poměrně jednoduchý proces. Dochází zde pouze k volání příkazu, který má na starost změnu maximální velikosti databázového souboru (viz příloha C). Kvótu lze jak zvýšit, tak i snížit. Problém je ale v tom, že kvóta po snížení nesmí být menší než je aktuální velikost databázového souboru. Z tohoto důvodu se provádí pouze zvyšování kvóty a kontroluje, zda se administrátor nesnaží kvótu nedopatřením snížit. Systém bude umožňovat kvótu pouze navyšovat. Implementace změny kvóty je k nahlédnutí v příloze C.

Záloha

Zálohování musí stejně jako vytváření mít určeno, kde se má daná záloha databáze vytvořit. Je zde použito podobného příkazu, který zjistí nastavenou implicitní zálohovací cestu. Poté se jen zvolí databáze, která se má zálohovat a provede její zálohování na cestu získanou z předešlého kroku.

4.2 Oracle

Jedná se o vyspělý multiplatformní databázový systém [8] s možnostmi zpracování dat na vysoké úrovni, vysokým výkonem a snadnou škálovatelností. Databázový systém Oracle spadá pod Oracle Corporation, což je jedna z hlavních společností, která se podílí na vývoji relační databáze či nástrojů pro správu databází. Oracle Database podporuje nejen standardní dotazovací jazyk SQL podle normy SQL92, ale také proprietární firemní rozšíření Oracle, programovací jazyk PL/SQL rozšiřující možnosti vlastního SQL. To umožňuje uživateli vytvářet uložené procedury, funkce, trigery apod. Také je zde podpora pro objektové databáze a databáze uložené v hierarchickém modelu dat (XML databáze).

Obrázek 4.5: *Logo Oracle Database*

4.2.1 Edice

Databázový systém Oracle je poskytován v několika verzích [14], kdy každá z nich je určena pro jiné účely. Aktuálně jsou na trhu následující verze:

Oracle Database Express Edition

Oracle Database Express Edition je jakýmsi vstupním bodem databáze stavěné na Oracle Database 11G R2. Je zdarma k vyvíjení, nasazení a distribuci. Je určena vývojářům Node.js, PHP, Java, .NET, XMl a Open Source aplikací. Tato verze je určena databázovým administrátorům, kteří potřebují bezplatnou startovní databází pro trénování a vývoj. Je zde omezení pouze na 1 procesor a databáze může mít maximálně 11 GB.

Oracle Database Standard Edition One

Databáze Oracle Standard Edition One je určena pro vývoj v malých organizacích a obchodních odděleních. Je omezena pouze na jeden server a maximálně dva sockety. Databáze Oracle Standard Edition One je dostupná pro širokou škálu systémů, jako Windows, Linux a Unix. Hlavními přednostmi této edice je rychlá instalace, ověřený výkon, spolehlivost, bezpečnost a škálovatelnost a také snadná správa s automatizovanými, samořídicími možnostmi.

Oracle Database Standard Edition

Tato edice je ideální pro středně velké společnosti. K dispozici je jedna připojitelná databáze s clustery Oracle Real Application Cluster zaručujícími dostupnost. Další výhodou je kompatibilita s databází Oracle Database Enterprise Edition, což znamená, že lze provést upgrade bez jakýchkoliv problémů. Je zde zlepšena kvalita služeb, díky výkonu, zabezpečení a dostupnosti podnikové třídy. Tato edice umožňuje využívat maximálně 4 sokety.

Oracle Database Enterprise Edition

Oracle Database Enterprise Edition poskytuje komplexní funkce pro zpracování nejnáročnějších transakcí, správu objemných dat a zatížení datových skladů. K dispozici je i funkce automatické optimalizace dat, která efektivně spravuje více dat, snižuje náklady na uložiště a zlepšuje výkon databáze. Bezproblémová integrace s produktem Oracle Enterprise Manager Cloud Control umožňuje správcům snadno spravovat celý životní cyklus databáze. Není zde limitace na sokety.

4.2.2 Administrace

Administrace Oracle je oproti předešlému Microsoft SQL Server náročnější [7]. Ačkoliv lze vytvářet databáze pro každého uživatele zvlášť, jedná se o časově náročný proces a proto je potřeba se podívat po jiném řešení. Řešením bude mít pouze jednu hlavní databázi a databázové účty budou tabulkovými prostory uvnitř ni. K těmto účtům se budou přiřazovat uživatelé podle určitých kritérií. Každý uživatel bude mít svůj předem daný prostor (daný kvótou) v tabulkovém prostoru USERS a k němu si bude přistupovat a obsluhovat ho. Oracle obsahuje pouze objekt User, který představuje konto uživatele, ke kterému se připojují jednotlivé tabulkové prostory a práva.

Připojení

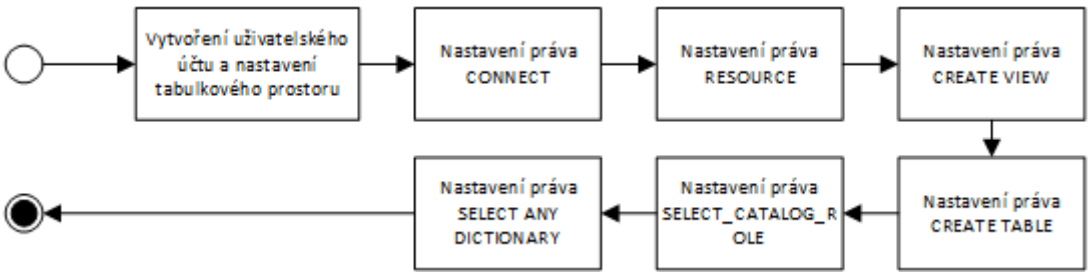
V rámci této aplikace je pro připojení využito ovladače Oracle Data Provider for .NET, který představuje optimalizovaný přístup k datům v databází Oracle pomocí jazyku .NET. Umožňuje vývojářům využívat pokročilé funkce databáze Oracle, včetně Real Application Cluster a XML. ODP.NET děla používání Oracle z prostředí .NET pružnější, rychlejší a stabilnější. ODP.NET obsahuje mnoho funkcí, které nejsou uvedeny v jiných .NET ovladačích jako vyrovnávání zatížení připojení či rychle připojení při selhání.

Vytvoření účtu

Při vytváření účtů se budou vytvářet tabulkové prostory v rámci zvolené databáze. Ve vybrané databázi bude využíván tabulkový prostor USERS. Postup bude takový, že se vytvoří uživatelský účet s vygenerovaným heslem, k němu se přidělí tabulkový prostor a nastaví kvóta. Nakonec se nastaví následující práva uživatele:

- CONNECT – právo umožňující se připojit k databázovému serveru
- RESOURCE – soubor práv umožňujících základní operace
- CREATE VIEW – právo umožňující vytvářet pohledy
- CREATE TABLE – právo umožňující vytvářet tabulky
- SELECT_CATALOG_ROLE – právo umožňující vykonávat dotazy v datovém slovníku
- SELECT ANY DICTIONARY – právo umožňující náhled do systémového katalogu

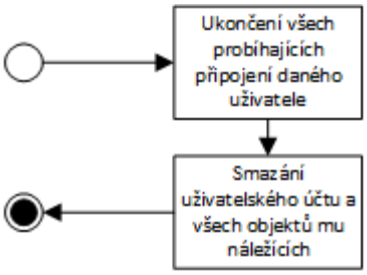
Vytvořený účet má práva umožňující připojení k oné hlavní databázi a vytváření objektů jako tabulky, triggery, funkce atd. Implementace vytváření účtů je k nahlédnutí v příloze D.



Obrázek 4.6: *Oracle – vytvoření databázového účtu*

Zrušení účtu

Mazání obnáší odstranění konkrétního uživatele, příkazem CASCADE zajistíme smazání všeho, co je s daným účtem spojené. Ale opět, nastával zde problém pří mazání uživatele, u kterého setrvává připojení. Řešení je obdobné, před samotným mazáním se naleznou všechny procesy náležící uživateli, kterého chceme smazat, a provede se jejich ukončení. Následně se může zavolat smazání samotného uživatelského účtu. Implementace mazání účtů je k nahlédnutí v příloze E.



Obrázek 4.7: *Oracle – smazání uživatelského účtu*

Restart hesla

V rámci restartu hesla je nutné provést odemknutí databázového účtu (předpokládá se, že uživatel si ho mohl zablokovat opakovaným zadáním nesprávného hesla) a následným nastavením nového hesla.

Změna hesla:

```
"ALTER USER " + login + " IDENTIFIED BY " + pass
```

Odemknutí účtu:

```
"ALTER USER " + login + " ACCOUNT UNLOCK", conn
```

Změna parametrů účtu - změna kvóty

U databázového systému Oracle je potřeba změnit kvótu pro vybraného uživatele na implicitním tabulkovém prostoru. Implementace změny kvóty je k nahlédnutí v příloze F.

Záloha

Vzhledem k existenci pouze jedné hlavní databáze není součástí práce řešení zálohování jednotlivých tabulkových prostorů.

4.3 PostgreSQL

PostgreSQL [19] je výkonný open-source objektově relační databázový systém. Má za sebou víc než 17 let aktivního vývoje a osvědčenou architekturu. Běží na všech běžných operačních systémech, včetně systému Linux, Max OS X, Solaris a Windows. Je plně kompatibilní s ACID, má plnou podporu na cizí klíče, pohledy a obsahuje procedurální nadstavbu. Podporuje také běžné datové typy jako INTEGER, VARCHAR, BOOLEAN, DATE a mimo jiné je zde i podpora pro velké binární objekty jako obrázky, zvuky či video. Obsahuje nativní programovací rozhraní pro C/C++, Java, .Net, Perl a jiné.

Obrázek 4.8: *PostgreSQL logo*

4.3.1 Edice

Na rozdíl od předešlých databázových systému, PostgreSQL nenabízí různé edice svého systému. Nabízí pouze jednu edici, která je jak již bylo psáno výše bezplatná. Jsou zde i určitá omezení, jako například maximální velikost tabulky 32 TB, maximální velikost řádku 1,6 TB apod.. Často je však uváděno, že PostgreSQL „nemá“ limity zaměřené na velikost [10].

4.3.2 Administrace

PostgreSQL umožňuje vytvářet efektivně databáze či uživatele bez nějaké větší prodlevy [10]. Jako u jiných databázových systému, i zde je potřeba k úspěšnému vytváření příkazů využívat dynamického SQL. PostgreSQL neumožňuje využívat dynamického SQL implicitně. Po delším bádání jsem narazil na možnost instalace rozšíření s názvem dblink. Jedná se o rozšíření PostgreSQL o možnost volání dynamického SQL za pomocí funkce dblink_exec. Funkce dblink_exec má dva parametry, *connstr*, který definuje připojení k databázi a poté *sql*, který představuje vytvořené dynamické SQL Je tedy potřeba zjistit název databáze, port a aktuálně přihlášeného uživatele (pomocí funkcí current_database, inet_server_port a current_user) a vytvořit tak připojovací řetězec, který následně vložíme do funkce dblink_exec. Pro snadnější nasazení PostgreSQL databázového systému jsou při vytváření procedur volány příkazy CREATE EXTENSION IF NOT EXISTS dblink, která zjišťují, zda již je požadované rozšíření nainstalováno, pokud ne, nainstalují se.

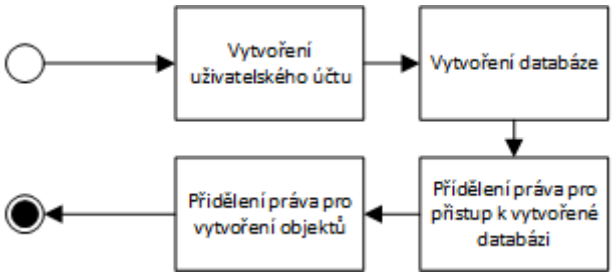
Připojení

V rámci této aplikace je pro připojení využito Npgsql. Jedná se o neoficiální PostgreSQL ovladač, který umožňuje připojení k databázovým serverům PostgreSQL prostřednictvím aplikací vyvíjených v .NET (Windows nebo ASP.NET).

Vytvoření účtu

Jelikož PostgreSQL umožňuje využívat sql_user, prvním krokem je vytvoření samotného uživatele s vygenerovaným heslem (viz příloha G). Dále je vytvářena databáze. Uživatele i databázi máme vytvořené, ale nejsou nastavena práva. Ta se nastaví voláním dvou příkazů pro nastavení přístupu k databázi a umožnění vytváření objektů. Implementace vytváření účtů je k nahlédnutí v příloze G.

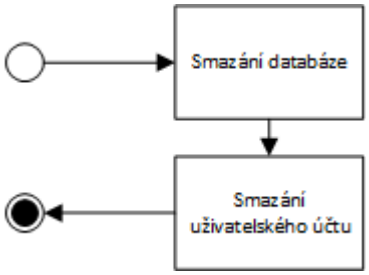
- CONNECT – právo připojení k databázi
- CREATE – právo umožňující vytváření objektů



Obrázek 4.9: *PostgreSQL – vytvoření databázového účtu*

Zrušení účtu

Smazání databázového účtu je velice jednoduché, není zde potřeba řešit nějaké ukončování stávajících připojení k databázím. Stačí v předem daném pořadí zavolat příkazy pro smazání databáze a následně pro smazání uživatelského účtu. Implementace mazání účtů je k nahlédnutí v příloze H.



Obrázek 4.10: *PostgreSQL – smazání databázového účtu*

Restart hesla

Jako v předchozích podkapitolách, jednoduchý příkaz, kdy je změněno přístupové heslo danému uživatelskému účtu.

Změna hesla:

```
"ALTER USER " + login + " WITH ENCRYPTED PASSWORD '" + pass + "';"
```

Změna parametrů účtu – změna kvóty

Velikost databáze nelze změnit pomocí databázového systému, lze to pouze pomocí operačního systému, a proto toto řešení není součástí práce.

Záloha

Záloha u PostgreSQL může být prováděna pouze pomocí příkazového řádku za pomocí aplikace pg_dump, která je součástí nadstavby příkazového řádku databázového systému PostgreSQL. Výstupem tohoto zálohování je SQL skript sloužící pro vytvoření databáze se všemi jejími náležitostmi.

4.4 MySQL

Jedná se o databázový systém, který byl vytvořený švédskou firmou Mysql AB [18], avšak nyní jej vlastní společnost Sun Microsystems, která dceřinou společností Oracle Corporation. Opět i tato databáze je multiplatformní, lze ji nainstalovat na Windows, Linux apod. Z počátku byl u MySQL kladen důraz na rychlost i přes nutnost zjednodušení (poskytovala jen jednoduché způsoby zálohování, do nedávna byla bez podpory pohledů, triggerů či uložených procedur).

Obrázek 4.11: *MySQL logo*

4.4.1 Edice

Databázové systém MySQL je na trhu dostupný v následujících verzích: [5][6]

MySQL Community Edition

Jedná se o bezplatnou edici MySQL databáze. V současné době se jedná o nejpopulárnější open source databázi. Její výhodou je obrovská aktivní komunita vývojářů.

MySQL Standard Edition

MySQL Standard Edition poskytuje vysoký výkon a škálovatelnost Online Transaction Processing (OLTP) aplikací. Poskytuje snadné použití, které dělá MySQL oblíbeným spolu s průmyslovým výkonem a spolehlivostí.

MySQL Enterprise Edition

MySQL Enteprise Edition obsahuje komplexní sadu pokročilých funkcí, nástroje pro správu a technickou podporu pro dosažení nejvyšší úrovně škálovatelnosti, bezpečnosti, spolehlivosti a dostupnosti.

MySQL Cluster Carrier Grade Edition

Objemy dat a uživatelské zatížení prudce rostou – je to dáno rostoucí dostupností a spolehlivosti internetu v celém světě, spadají pod to globální komunity, sociální sítě či vysoko rychlostí mobilní připojení. Hlavními kritérii této verze jsou tedy škálování I/O operací, nízká přístupová doba nebo maximální dostupnost 24x7.

4.4.2 Administrace

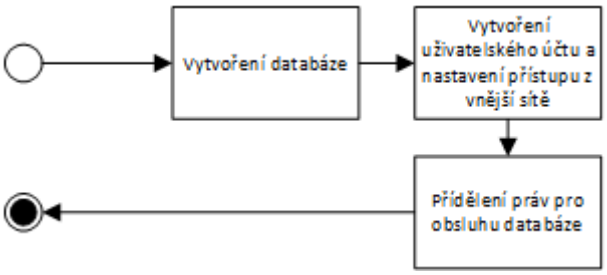
MySQL umožnuje připojení pouze svým uživatelům pomocí tzv. MySQL Accounts [5]. Nepracuje vůbec s uživatelskými účty operačního systému. Standardní vytváření uživatelského účtu za pomocí jeho názvu a hesla je zde rozšířeno o nastavení, odkud se může daný uživatel přihlásit (localhost, internet), bez těchto nastavení nelze uživatelský účet vytvořit.

Připojení

MySQL také poskytuje ovladače pro velké množství vývojových jazyků. Všechny jsou k dispozici ke stažení ze stránek produktu [27]. K připojení je využito Connector/Net. Jedná se o ovladač plně využitelný pro vývoj .NET aplikací pod MySQL. Od verze 6.7 se nabízí jako doplněk MySQL for Visual Studio.

Vytvoření účtu

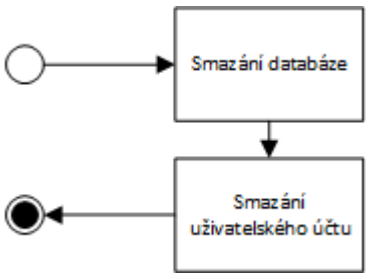
Vytváření databázového účtu zahrnuje 3 kroky. Prvním je vytvoření samotné databáze. Dále se volá příkaz pro vytvoření uživatele s heslem zahrnující i udělení práva pro přístup z vnější sítě. A jako poslední se přidělují práva pro obsluhu právě vytvořené databáze (SELECT, INSERT, CREATE, DROP atd.). Implementace vytváření účtů je k nahlédnutí v příloze I.



Obrázek 4.12: *MySQL – vytvoření databázového účtu*

Zrušení účtu

Mazání je obdobné jako u PostgreSQL, i zde se nemusí řešit existující připojení k databázím, které by blokovaly jejich mazání. Je tedy potřeba smazat databází a poté samotného uživatele. Implementace mazání účtů je k nahlédnutí v příloze J.



Obrázek 4.13: *MySQL – smazání databázového účtu*

Restart hesla

Restart hesla je vyvolán příkazem:

Změna hesla:

```
"SET PASSWORD FOR '" + login + "' = PASSWORD('" + pass + "')"
```

Změna parametrů účtu - změna kvóty

U databázového systému MySQL nedefinuje při vytváření databáze její kvótu, proto zde není možnost její změny.

Záloha

Záloha může být prováděna pouze z příkazového řádku za pomocí aplikace mysqldump, která je součástí nadstavby příkazového řádku databázového systému MySQL. Výstupem tohoto zálohování je opět SQL skript sloužící pro vytvoření databáze se všemi jejími náležitostmi.

4.5 IBM DB2

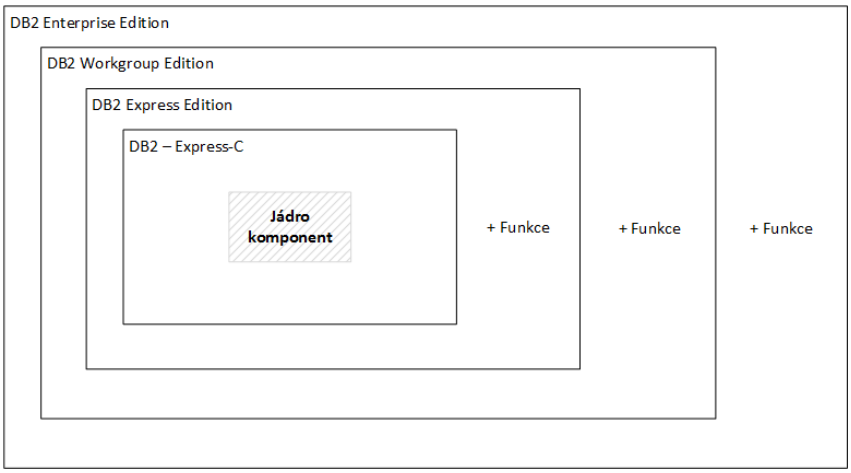
IBM DB2 [1] je relační databázový systém společnosti IBM. Jedná se multiplatformní databázi s pokročilými možnosti kontroly dat, jejich využívání a ochrany. Je nabízen v několika edicích, které se liší nabízenými funkcemi a také výkonem určeným omezením.

Obrázek 4.14: *Logo DB2*

4.5.1 Edice

Z obrázku 4.15 vyplývá, že všechny edice IBM DB2 serverů sdílí stejné jádro komponentů a další rozšíření těchto edicí je realizování dodáním balíčků. Těmito balíčky si koncový uživatel může určit edici, kterou potřebuje. Například DB2 Express-C lze považovat za identický jako DB2 Express Edition s tím rozdílem, že DB2 Express Edition obsahuje určité komponenty navíc. Uživatel si může vybrat určitou edici, a pokud mu bude něco chybět, stačí upgradovat na vyšší edici.

Znázorněná hierarchie edicí také poukazuje na to, jak je jednoduché upgradovat z nižší edice na edici vyšší. Pokud uživatel uvažuje o rozšíření své dosavadní edice, nemusí nijak řešit přechod již vyvinutých aplikací na dosavadní edici IBM DB2 serveru. Toto zajišťuje právě zmiňované jádro komponent, které mají uživatelovy edice stejné. Dále také platí, že již použitá syntaxe je stejná, a není ji potřeba s příchodem vyšší edice měnit.



Obrázek 4.15: *Hierarchie edicí IBM DB2 serverů*

DB2 Express-C

DB2 Express-C je server dostupný ke stažení bez poplatku. Písmeno C jeho názvu značí Comunity (Komunita), což znamená, že tato edice je vhodná pro komunitu databázových nadšenců. Jedná se o ideální server pro malé podnikání a vývojáře, kteří vyvíjejí své databázové aplikace klientům. DB2 Express-C neukládá limity na počty instancí na server nebo počet uživatelů. Avšak má určitá omezení týkající se zatížení systému. Je dostupný pro Windows, Linux i Mac. Server může využívat maximálně 16 GB paměti a využívat maximálně 2 procesorová jádra.

DB2 Express Edition

DB2 Express Edition je ideální pro podnikání, kde uživatelé požadují datový server, ale nemají dostatek in-house databázových zkušeností. Tato edice poskytuje stejnou podporu jako všechny ostatní edice a také má jednoduchou instalaci.

DB2 Workgroup Server Edition

DB2 Workgroup Server Edition je datový server s funkcionalitou vhodnou pro jeden databázový server, pracovní skupinu nebo středně velké podnikatelské prostředí. V porovnání s funkcemi nabízenými v DB2 Express Edition, DB2 Workgroup Edition umožňuje rozdělovat tabulky a podpora platforem je rozšířena o AIX, Solaris a HP-UX. Na rozdíl od předešlých edicí, tato dokáže lépe využít výkonu systému, konkrétně maximálně 128 GB paměti a 16 procesorových jader.

DB2 Enterprise Server Edition

DB2 Enterprise Server Editon je nabízen pro střední či velké podniky. Poskytuje velký výkon, škálovatelnost, dostupnost a rozšiřitelnost funkcí, které ji dělají nejvíce oblíbenou volbou pro mnoho podniků. Je nabízen pro Linux či Windows a nemá žádné limity na systémový výkon.

4.5.2 Administrace

V porovnání s ostatními zmiňovanými systémy zabral tento databázový systém při vývoji nejvíce času [1]. Jedná se totiž o trochu odlišný systém. Prvním zásadním rozdílem je, že DB2 nemá

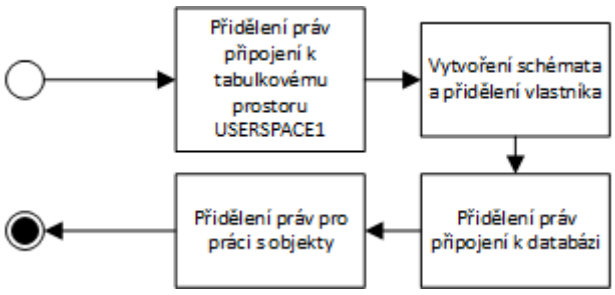
správu vlastních uživatelských účtů a je silně vázán na uživatelské účty operačního systému, na kterém běží. Správa databázových účtů na databázovém serveru by tedy měla vycházet ze správy uživatelských účtů v operačním systému. Druhý rozdíl se objevuje u správy samotných databází, které nelze vytvářet, respektive mazat pomocí jazyka SQL a jeho procedurální nadstavby. Pro tyto účely má databázový systém DB2 samostatnou sadu konzolových aplikací. První problém řeší školní LDAP systém. Každý uživatel školy má přidělen svůj login, který představuje záznam v adresáři LDAP. Pomocí tohoto účtu bude uživatel přistupovat ke svému databázovému prostoru. Druhý problém lze vyřešit tak, že se nebude každému uživateli vytvářena samostatná databáze, pouze se bude vytvářet schéma na jedné hlavní databázi, které mu bude přiděleno.

Připojení

Pro připojení je využito IBM Data Server Driver Package. Jedná se balíček ovladačů sloužící pro .NET aplikace, který je k dispozici ke stažení ze stránek společnosti IBM [15].

Vytvoření účtu

U IBM DB2 je uživatelský účet vždy stejný, nemění se spolu s názvem databázového účtu. Nastává zde problém, kdy by si uživatel chtěl vytvořit objekt (například tabulku) s identickým názvem pod jedním prostorem. Jelikož se přihlašuje pod jediným uživatelským účtem, systém mu to nedovolí a zahlásí chybu ve smyslu, že již existuje objekt se stejným názvem v daném tabulkovém prostoru. Proto je potřeba uživateli vytvořit schéma zaručující řešení tohoto problému. Uživatel bude mít přístup pouze k danému schématu a žádnému jinému (které nevlastní). Implementace vytváření účtů je k nahlédnutí v příloze K.

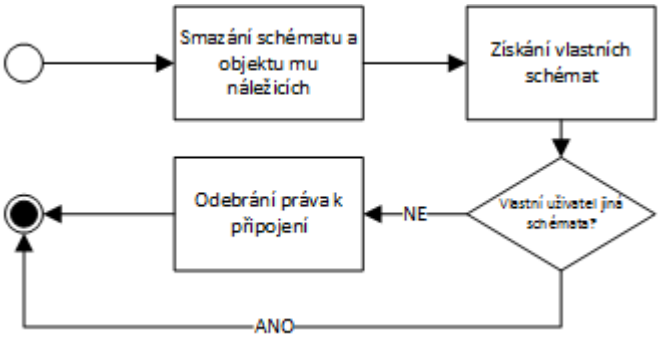


Obrázek 4.16: *IBM DB2 – vytvoření databázového účtu*

Zrušení účtu

Smazání databázového účtu se skládá z následujících kroků. Nejprve příkaz ADMIN_DROP_SCHEMA, který zajistí smazání všech objektů náležících danému schématu (podobně jako CASCADE). Jelikož je zde uživatelský účet Windows, nelze jej smazat, lze mu pouze odebrat právo k připojení k databázi. Jelikož je uživatel reprezentován účtem mimo databázový systém, v případě, že bychom odebrali právo při každém mazání schématu, by mohlo dojít k případu, že uživatel má v systému i jiné schéma náležící například alternativnímu názvu a znemožnili bychom mu tím přístup k databázi. Proto je po smazání schématu kontrolována existence jiných vlastnických

práv k jiným schématům. Pokud by nebyly nalezeny jiná schémata, která uživatel vlastní, je mu odebráno přístupové právo k samotné databázi. Implementace mazání účtů je k nahlédnutí v příloze L.



Obrázek 4.17: *IBM DB2 – smazání databázového účtu*

Restart hesla

Jak vyplývá z popisu, u IBM DB2 není možno řešit reset hesla, jelikož je dáno LDAP záznamem.

Změna parametrů účtu - změna kvóty

Vzhledem k použití schémat při vytváření účtů zde není možnost změny kvóty.

Záloha

Vzhledem k existenci pouze jedné hlavní databáze není součástí práce řešení zálohování jednotlivých schémat.

4.6 Shrnutí kapitoly

Dalo by se konstatovat, že všechny databázové systémy, se kterými jsem pracoval, jsou postaveny tak, aby nabídly podporu pro velké množství různých programovacích jazyků. Čili není pravidlem, že Oracle Database, která je vyvíjena především společností Oracle a pod kterou spadá i jazyk Java, bude poskytovat ovladače pouze pro tento jazyk Java, ba naopak připojení je zajištěno i pro jazyk .NET. Co musím z vlastní zkušenosti poznamenat je, že co se týče zavedení ovladačů a jejich použití, mají všechny mnou zmiňované systémy stejnou logiku. Je to dáno tím, že všechny jejich ovladače jsou postaveny na stejných rozhraních (IConnection, ICommand, IDataSource, ITransaction atd.). S tím souvisí velice jednoduchý přechod mezi jednotlivými servery. Není potřeba složitě studovat, jak daný ovladač funguje, protože se všechny obsluhují stejným způsobem. V dohledné době se očekává nasazení i jiných databázových systémů a vize je taková, že pro jejich nasazení bude potřeba pouze malých úprav k jejich úspěšnému zavedení do systému DB Manager.

5 Analýza

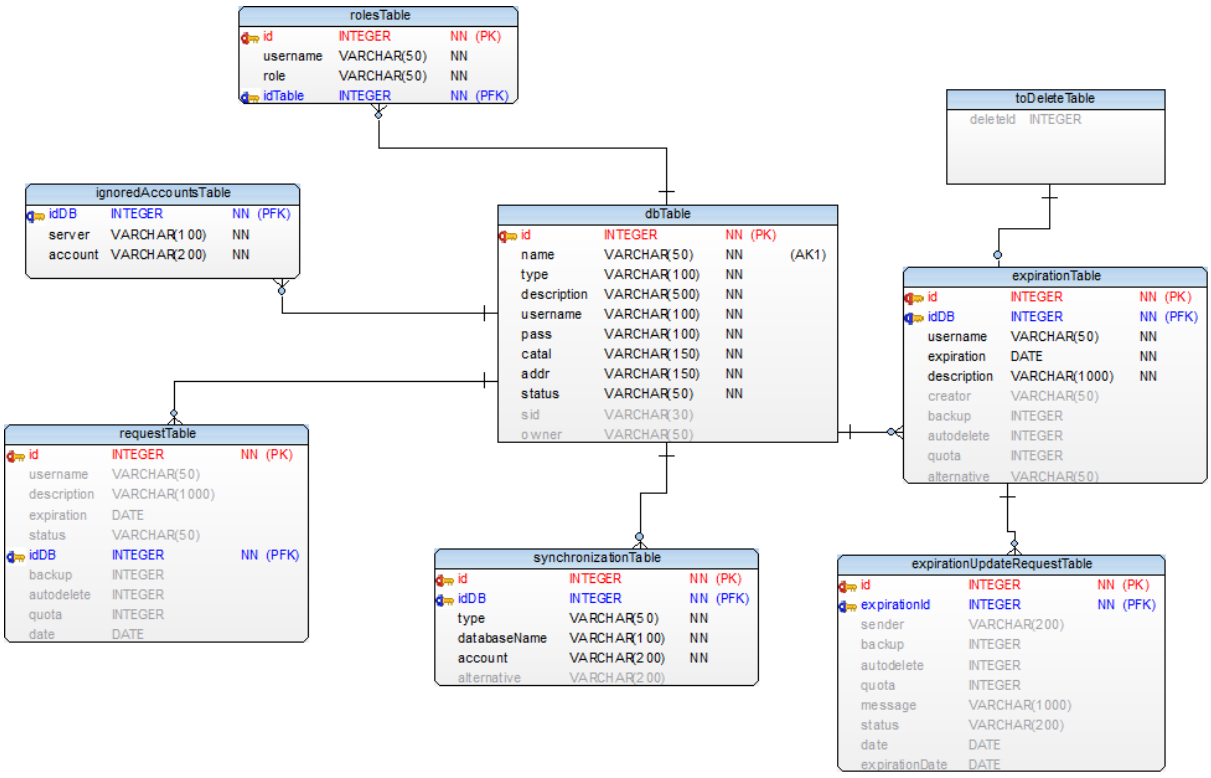
5.1 Vstupy systému

Systém bude sloužit pro správu databázových účtů. Spolu s nimi je potřeba si ukládat jejich záznamy. Ovšem spravovat tyto účty budou moci pouze předem určení uživatelé. To znamená, že ne každý si bude moci sám vytvářet účty, jaké potřebuje. Uživatelé s minimálními právy si nicméně budou moci poslat žádost o účet, o které rozhodne jeden z administrátorů. Všechny tyto žádosti, záznamy o účtech, atd. se budou ukládat do databáze.

5.2 Datová analýza

Pro správný chod je nutné ukládat si určité informace o spravovaných databázových systémech a databázových účtech jim náležících.

5.2.1 Datový model



Obrázek 5.1: *Datový model*

5.2.2 Popis tabulek

Následující seznam uvádí, k čemu slouží jednotlivé tabulky:

- **dbTable** – tabulka uchovávající informace o spravovaných databázových systémech,
- **expirationTable** – tabulka databázových účtů náležících databázovým systémům,

- **requestTable** – tabulka žádostí o vytvoření databázových účtů,
- **expirationUpdateRequestTable** – tabulka evidující žádost o změnu parametrů databázových účtů,
- **toDeleteTable** – tabulka označující ty účty, které jsou připraveny ke smazání,
- **synchronizationTable** – tabulka označující konflikty databázových účtů,
- **ignoredAccountsTable** – tabulka účtu, které mají byt synchronizací ignorovány,
- **rolesTable** – tabulka evidující role systému DB Manager.

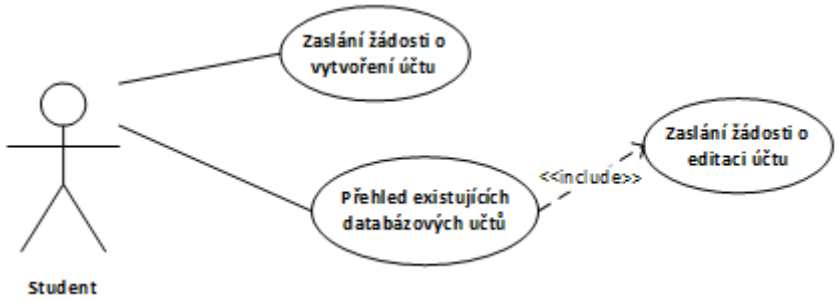
5.3 Uživatelské role

Systém je určen jak studentům, tak i pedagogům. Každý má ale jiná práva a proto je potřeba je od sebe rozlišit. Proto jsou zavedeny 3 role uživatelů:

- Student
- Databázový administrátor (dále DB Admin)
- Systémový administrátor (dále SYS Admin)

Student

Student obsluhuje databázové účty mu náležící. Primární funkcí je možnost poslat žádost o vytvoření databázového účtu. Má k dispozici ale i přehled o svých existujících účtech nad danými databázovými servery. Tyto účty mají určité parametry a ty lze na základě poslání žádosti měnit (o schválení žádosti o změnu parametrů účtu rozhoduje DB nebo SYS Admin).



Obrázek 5.2: *UseCase diagram – role „student“*

Databázový administrátor

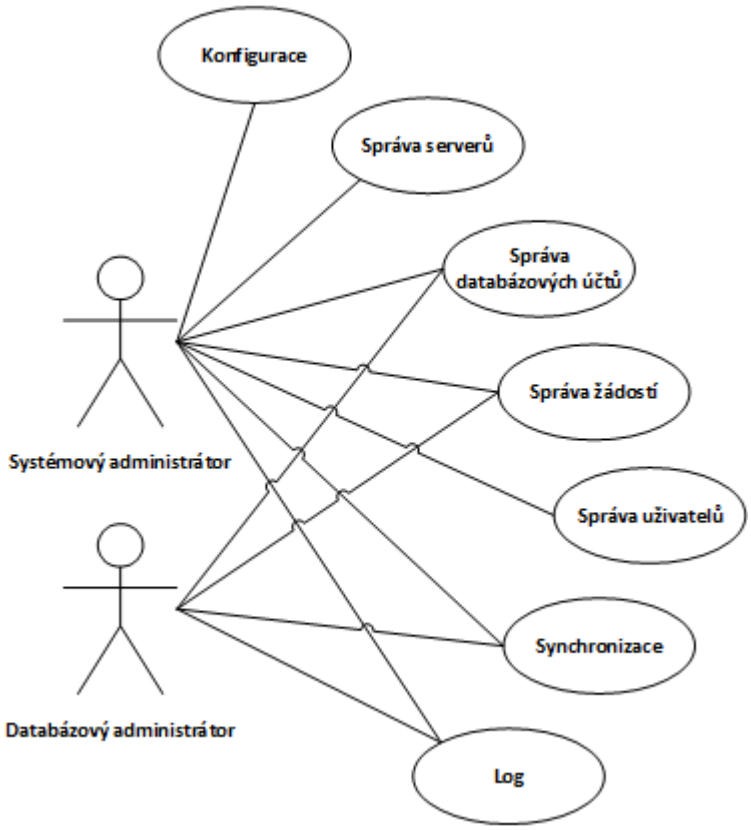
Hlavním úkolem **DB Admin** je správa databázových účtů a správa žádostí nad databázovými systémy, které jsou mu přiděleny. SYS Admin i DB Admin mají přístup do sekce Log, kde se ukládají všechny operace prováděné v systému DBManager a také vzniklé chyby.

Systémový administrátor

Systémový administrátor má přístup ke všem funkcím systému. Jeho hlavním úkolem je obstarávat všechny nastavené databázové servery a celkově řídit chod systému DB Manager. Má právo spravovat databázové, všechny jejich informace o připojení, dostupnosti a atd. Dále má SYS

Admin k dispozici právo pro konfiguraci systému, což znamená hlavní nastavení systému DB Manager jako délka hesla, email odesílatele, nastavení plánovače atd.

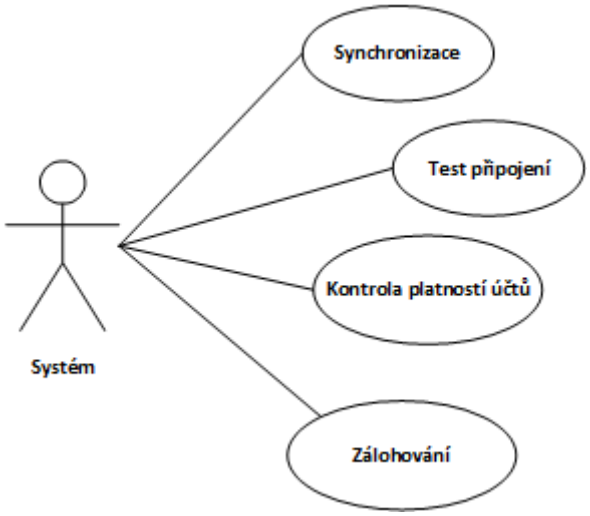
Další funkce má SYS Admin identické jako DB Admin s tou výjimkou, že SYS Admin spravuje žádosti a objekty všech databázových serverů, kdežto DBAdmin se stará pouze o ty účty, které jsou umístěny na serverech mu přidělených.



Obrázek 5.3: *UseCase diagram – správa databázových účtů*

Systém

Poslední rolí, která však není součástí webového rozhraní, je systémová role, která provádí automatické procesy spuštěné plánovačem systému Windows.



Obrázek 5.4: *UseCase diagram - systém*

5.4 Funkční analýza

V této kapitole budou podrobněji rozebrány složitější funkce systému vzhledem k jejím rolím.

5.4.1 Zaslání žádosti o vytvoření účtu

Tabulka: requestTable

Zodpovědnost: student

Funkce:

a) Zaslání žádosti

5.4.2 Přehled existujících databázových účtů

Tabulka: expirationstable, expirationUpdateRequestTable

Zodpovědnost: student

Funkce:

a) Reset hesla
b) Zaslání žádosti o změnu parametrů databázového účtu
c) Smazání databázového účtu

5.4.3 Konfigurace

Tabulka:

Zodpovědnost: systémový administrátor

Funkce:

a) Uložení konfigurace
b) Nastavení plánovače

5.4.4 Správa serverů

Tabulka: dbTable

Zodpovědnost: systémový administrátor

Funkce:

a) Přidání serveru
b) Odebrání serveru
c) Změna parametrů serveru
d) Vytvoření procedur

5.4.5 Správa databázových účtů

Tabulka: expirationsTable

Zodpovědnost: systémový administrátor, databázový administrátor

Funkce:

a) Vložení databázového účtu
b) Filtrace databázových účtů
c) Smazání databázového účtu
d) Změna parametrů databázových účtů
e) Záloha databázových účtů
f) Reset hesla

5.4.6 Správa žádostí

Tabulka: requestTable, expirationUpdateRequestTable

Zodpovědnost: systémový administrátor, databázový administrátor

Funkce:

a) Schválení žádosti
b) Zamítnutí žádosti
c) Úprava žádosti
d) Odeslání dotazu

5.4.7 **Správa uživatelů**

Tabulka: rolesTable

Zodpovědnost: systémový administrátor

Funkce:

a) Přidání role
b) Odebrání role

5.4.8 **Synchronizace**

Tabulka: ignoredAccountsTable, synchronizationTable

Zodpovědnost: systémový administrátor, databázový administrátor

Funkce:

a) Vložení ignorovaného účtu
b) Odebrání ignorovaného účtu
c) Spuštění synchronizace
d) Smazání konfliktních účtů
e) Vložení konfliktních účtů

5.4.9 **Log**

Tabulka:

Zodpovědnost: systémový administrátor, databázový administrátor

Funkce:

a) Zobrazení detailu záznamu
b) Filtrace záznamů

6 Použité technologie a nástroje

6.1 Prezentační vrstva

6.1.1 ASP.NET

Pro vývoj byla vybrána platforma ASP.Net a to z několika důvodů. Mezi ně patří jak předešlé zkušenosti s touto platformou, tak již zaběhlé informační systémy na dbedu.cs.vsb.cz, které byly rovněž vyvíjeny za pomocí technologie ASP.NET. Další aspektem, bylo plánované vyvíjení hlavní komponenty pomocí jazyku C# a její následné propojení jak s webovým rozhraním pro obsluhu databázových systémů, tak i s aplikací pro provádění automatických operací. Také vytvoření komponent, které by byly přenositelné mezi ostatními webovými portály databázové skupiny, hrálo svou roli. Jednalo se o sjednocení přihlašovacího systému, sjednocení formy pro posílání emailů, atd. Pro vývoj byl použit Microsoft Visual Studio 2013 a .NET Framework 4.5.

6.2 Datová vrstva

Jako každá jiná aplikace si i systém DB Manager potřebuje ke svému chodu ukládat data. Tato data představují jak určitá nastavení systému, tak i záznamy vytvořené za jeho běhu. Vzhledem k předešlým zkušenostem se SQLite a jeho využívání při vývoji Android aplikací, bylo využito právě tohoto systému.

Pro uložení nastavení aplikace je využíváno již zavedeného prvku WebConfig technologie ASP.NET. Jedná se o nastavování hodnot ve formě klíč = hodnota (key = value). Tato nastavení lze za běhu aplikace jak měnit, tak k nim i přistupovat z jiných komponent. Ve skutečnosti se jedná o XML soubor, který obsluhuje již zavedená třída, a proto není potřeba vymýšlet novou komponentu.

6.2.1 SQLite

SQLite [11][12] je relační databázový systém obsažený v relativně malé knihovně napsané v jazyku C. Na rozdíl od databází založených na principu klient-server, kde je databázový server spuštěn jako samostatný proces, je SQLite pouze malá knihovna, která se přilinkuje k aplikaci. Každá databáze je uložena v samostatném souboru, kde se data ukládají za použití jednoduchého primárního klíče do stejně velkých bloků. SQLite využívá hašovacích technik pro rychlý přístup datům při vyhledávání podle klíče.

Obrázek 6.1: *Logo SQLite*

Konektivita

SQLite poskytuje stejně jako ostatní databázové systémy ovladače potřebné k připojení k databázovému souboru v mnoha jazycích či systémech (Windows, Mac OS X a Linux). Připojení je

realizováno pomocí SQLite ODBC ovladače, kde se namísto adresy serveru uvádí cesta k databázovému souboru.

Podporované jazyky

- C / C++
- C#
- Delphi
- Java
- PHP
- Python
- Perl
- Ruby

6.3 Databázoví klienti

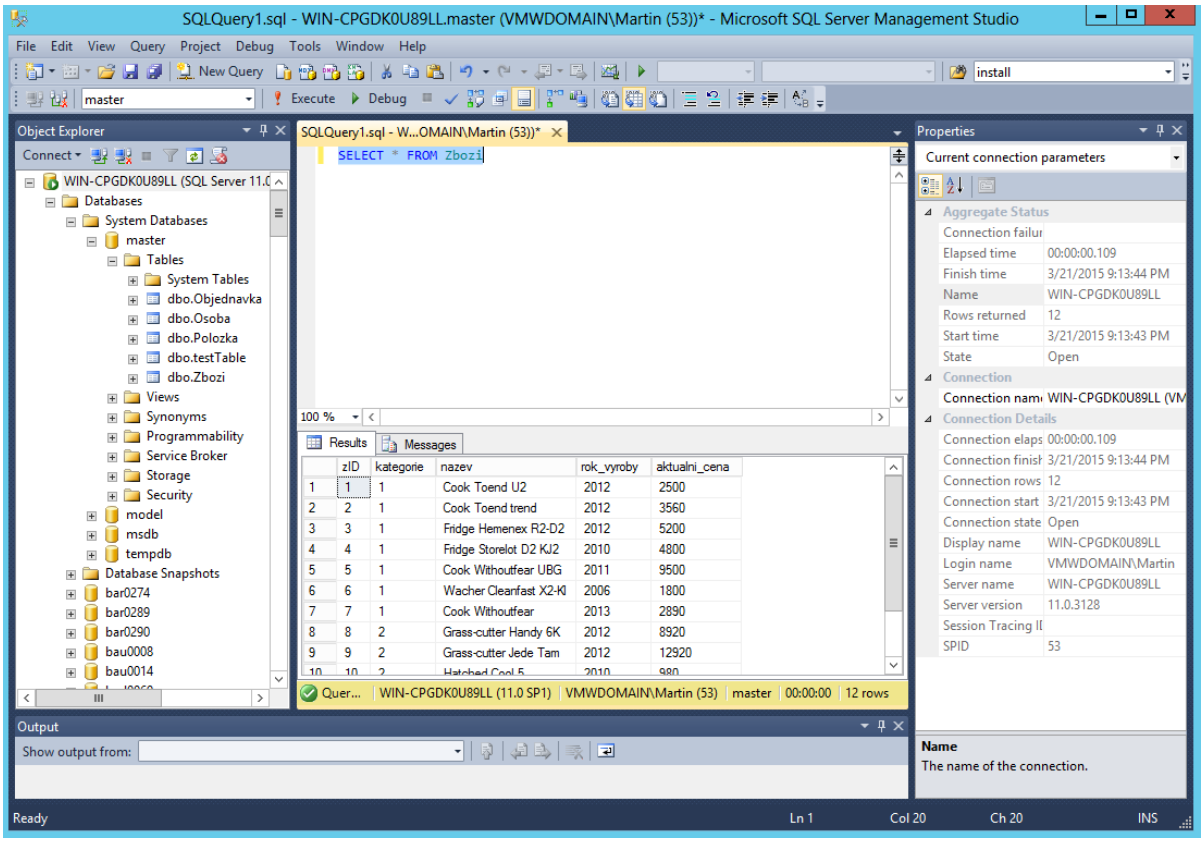
Ačkoliv se dá pro správu databázových systému použit univerzální klient, rozhodl jsem se využít oficiálních klientů dodávaných spolu s databázovými systémy. Dle mého názoru oficiální klienti poskytují daleko lepší přehled o administraci systému. Jak bylo uvedeno v předešlé kapitole, každý databázový systém má jinak řešenou svou administrační část a pro tyto účely poslouží nejlépe oficiální klienti.

6.3.1 SQL Server Management Studio

SQL Server Management Studio [16] je do jisté míry považováno za základ při administraci SQL Serveru. Poskytuje celou řadu funkcí pro správu serveru pomoci grafického uživatelského rozhraní. Je podobný Visual Studiu, kombinuje bezpočet funkčních prvků.

Zde je rychlý přehled funkcí, které SQL Server Management Studio poskytuje:

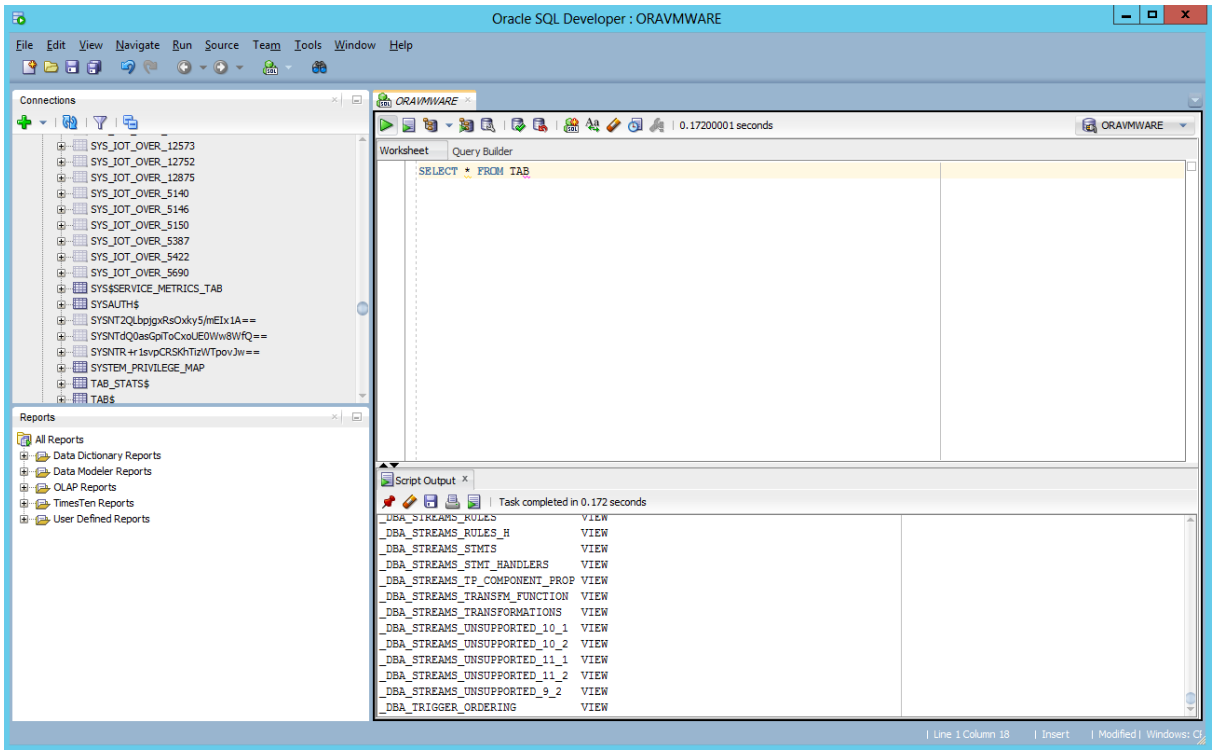
- vytváření, editace a mazání databází a databázových objektů,
- dotazování se nad databází pomocí jazyku T-SQL,
- správa plánovaných úloh jako například zálohování,
- přehled aktuálních aktivit, kdo je přihlášen, které objekty jsou zamknuty,
- nastavení bezpečnosti, jako jsou role, uživatelé,
- vytváření a správa full-textových vyhledávacích katalogů.



Obrázek 6.2: *SQL Server Studio*

6.3.2 Oracle SQL Developer

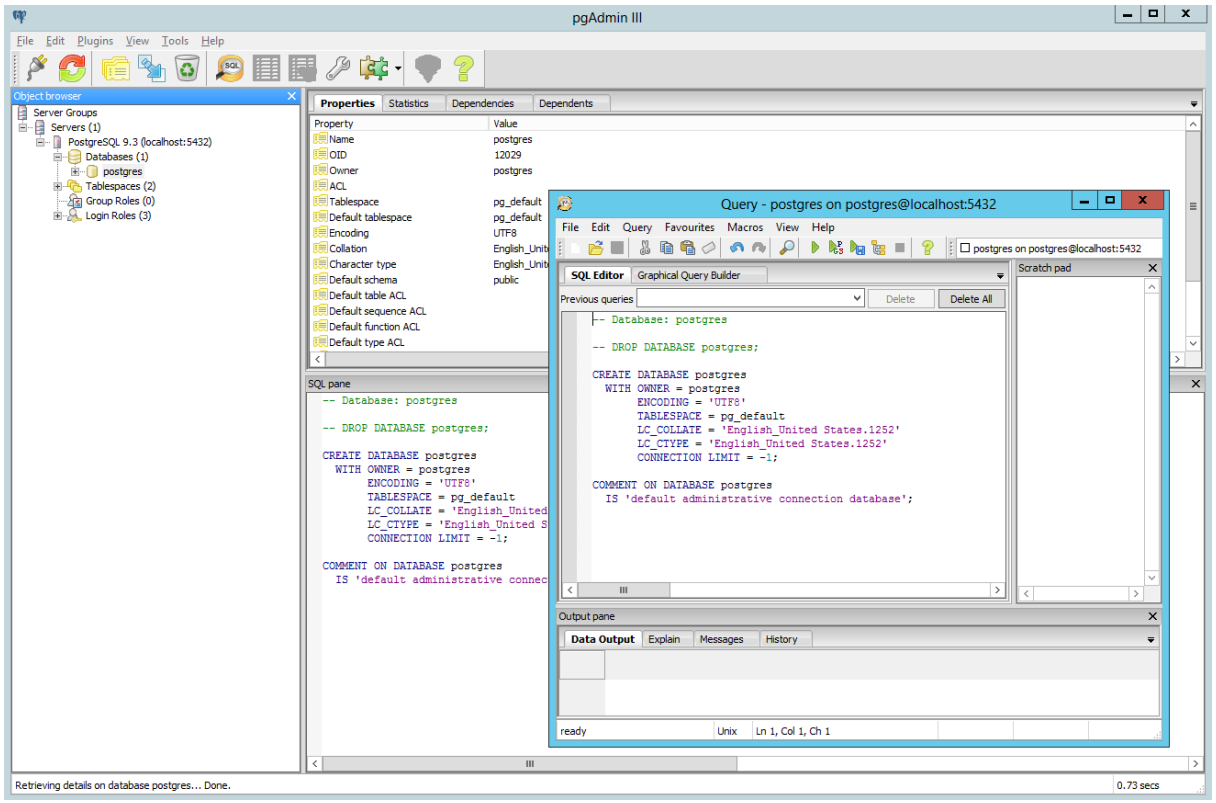
Oracle SQL Developer [14] je IDE Oracle databázového systému. Jedná se o bezplatné grafické rozhraní umožňující administraci databáze. Hlavním úkolem Oracle SQL Developer je uživateli ulehčit čas a maximalizovat efektivnost jeho práce. SQL Developer podporuje Oracle Database 10g, 11g a 12c a funguje na všech operačních systémech, které podporují Java. Hlavními prvky jsou správa uživatelů a jejich rolí, správa záloh, data flow diagrams (DFD), spouštění a ladění skriptů, přehled o uložených procedurách, funkcích, atd.



Obrázek 6.3: *Oracle SQL Developer*

6.3.3 PgAdmin III

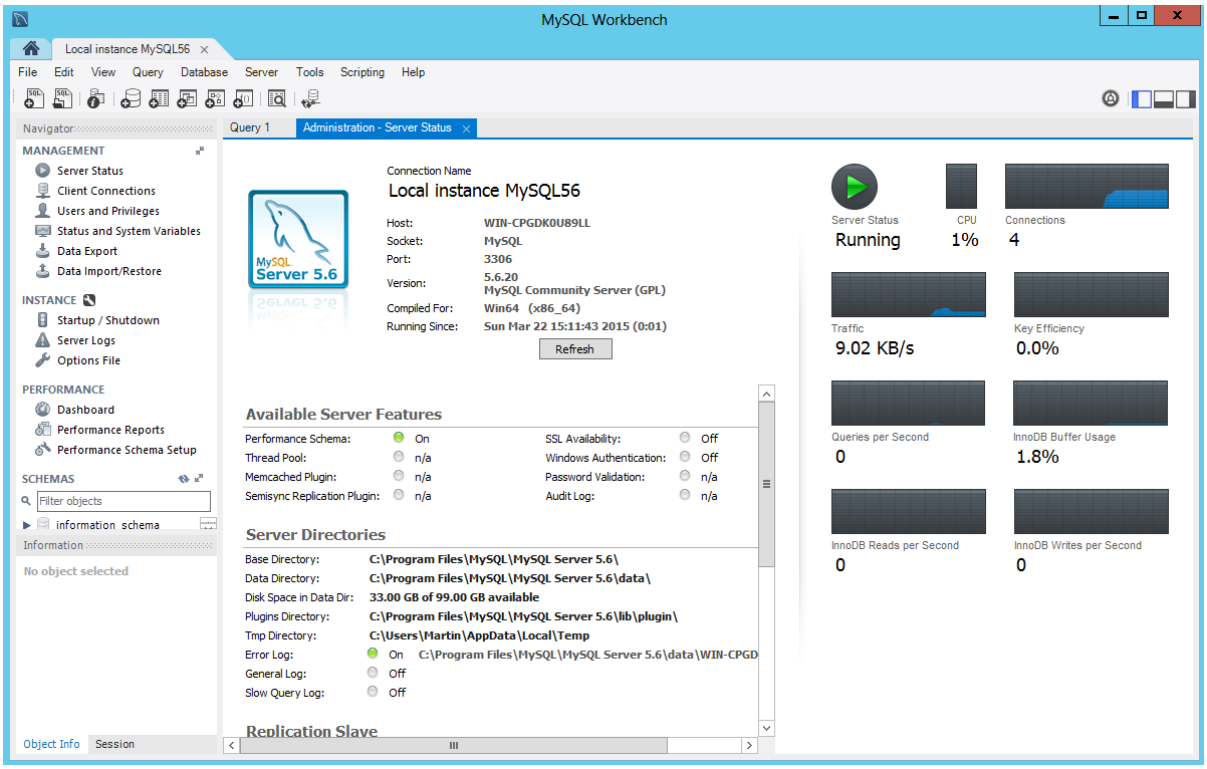
PgAdmin III [10] je populární a dobře vybavená administrační open-source aplikace pro správu PostgreSQL. Aplikace je nabízena pro Linux, Solaris, Max OS X a Windows. PgAdmin III má za úkol uspokojit potřeby uživatelů od psaní jednoduchých dotazů až po vývoj komplexních databází. Grafické rozhraní dělá správu jednoduchou a podporuje všechny funkce PostgreSQL. Výhodou je také zvýrazňování syntaxe kódu. Připojení k serveru je realizováno pomocí TCP/IP nebo Unixovými sokety. PgAdmin III je vyvíjen komunitou expertů PostgreSQL z celého světa a tím pádem je dostupný ve velké škále jazykových mutací.



Obrázek 6.4: *pgAdmin III*

6.3.4 MySQL Workbench

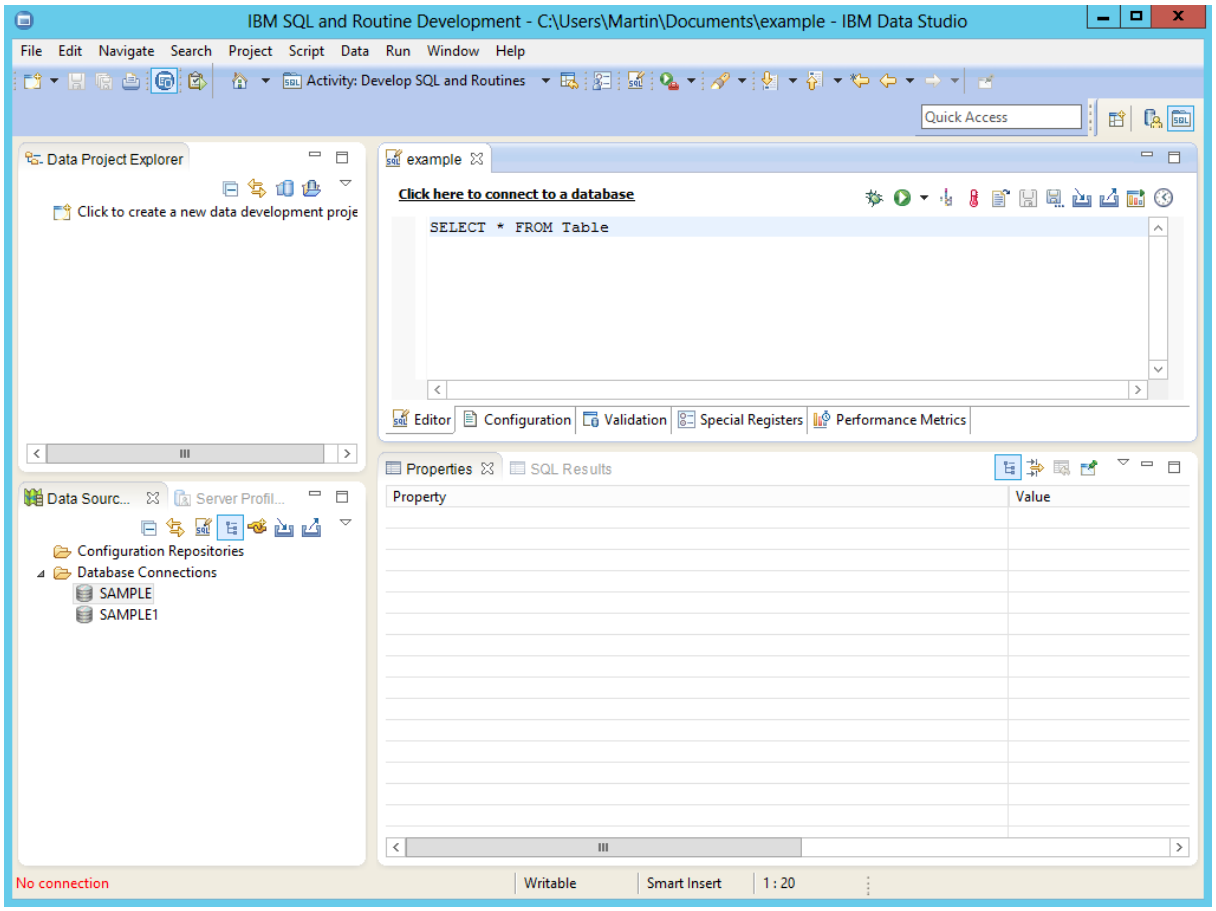
MySQL Workbench [18] je vizuální nástroj sloužící databázovým architektům, vývojářům či administrátorům. Je zde k dispozici možnost modelování databází, vývoj SQL a další nástroje určené pro konfiguraci systému, správu uživatelů či zálohování. MySQL Workbench je k dispozici pro Windows, Linux a Mac OS X.



Obrázek 6.5: *MySQL Workbench*

6.3.5 IBM Data Studio

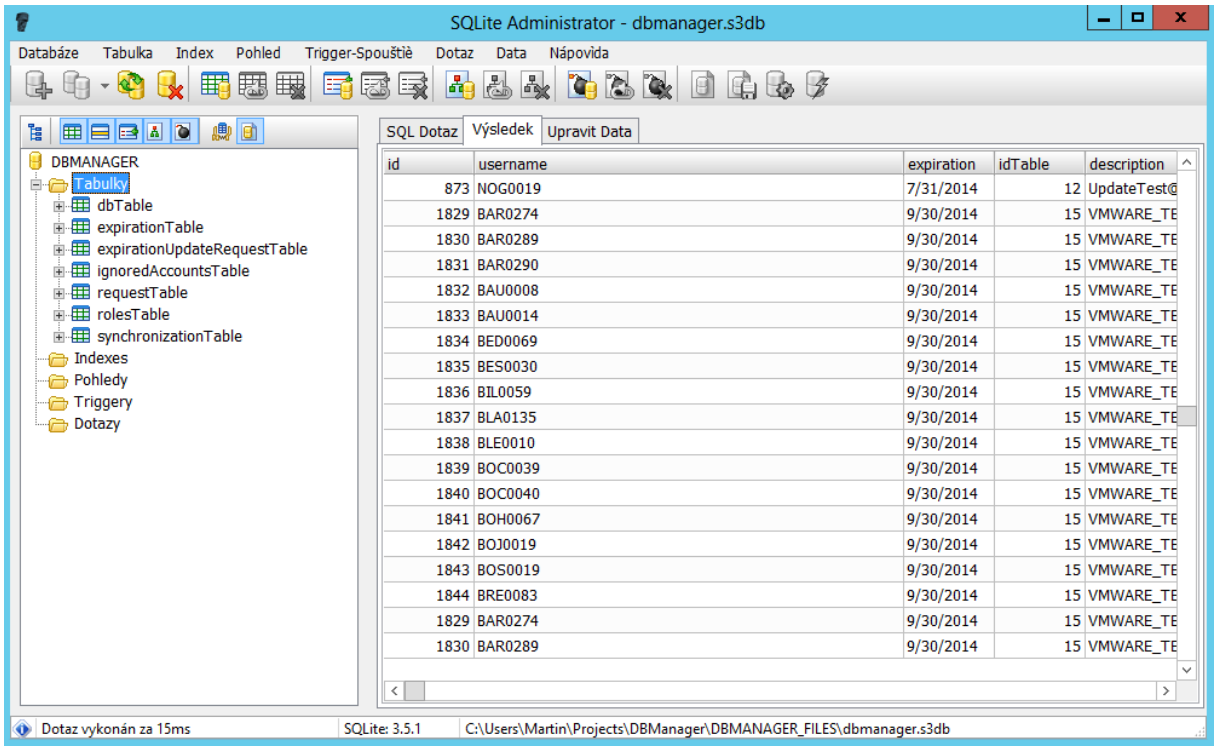
Jedná se o nástroj určený ke správě databází [1], vývoji XQuery, SQL skriptů, uživatelsky definovaných funkcí a uložených procedur. Nástroj je založený na Eclipse, jako většina nástrojů od IBM. Studio umožňuje debutovat vyvíjené skripty a také má k dispozici nástroj pro práci s diagramy fyzického modelu dat (PDM) sloužící k lepšímu přehledu mezi vztahy a tabulkami. IBM Data Studio nahrazuje DB2 nástroje (Control Center a Command Editor), které se v současné době již nevyužívají. IBM Data Studio je dostupné jak pro Windows, tak i pro Linux.



Obrázek 6.6: *IBM Data Studio*

6.3.6 SQLite Administrator

Společnost SQLite neposkytuje implicitně uživatelské rozhraní pro přehlednější administraci databáze. Avšak existují i aplikace vyvíjené nezávislými vývojáři jako třeba SQLite Administrator. Jedná se o aplikaci od skupiny Orbmu2k. Jejím záměrem je poskytnout jednoduchou administraci databázového souboru. Je zde k dispozici přehled tabulek, filtrace záznamů v tabulkách, vytváření tabulek pomocí jednoduchého rozhraní nabízející nastavení datových typů, implicitních hodnot atd.



Obrázek 6.7: *SQLite Administrator*

7 Implementace systému

7.1 Webové rozhraní DB Manager

Webové prostředí systému DB Manager poskytuje jednoduchý přístup ke všem funkčnostem pro správu jednotlivých databázových systémů. Jeho design je sjednocený s ostatními systémy portálu dbedu.cs.vsb.cz (viz obrázek 7.2). Úkolem tohoto webového prostředí je poskytnout uživateli jednoduchý přehled spravovaných databázových systémech, databázových účtech, žádostech, atd. Také poskytuje jednoduchou manipulaci s rozhraním pro administraci (viz kapitola 3.4) a tedy nekomplikovanou administraci všech připojených databázových serverů.

Obrázek 7.1: *Logo systému DB Manager*

Obrázek 7.2: *Webové rozhraní DB Manager*

7.1.1 Menu

Role Student

Poslat žádost – stránka určená studentům k odeslání žádosti o vytvoření účtu (funkce 5.4.1/a). Každý databázový systém poskytuje 4 stavy databázového účtu:

- aktivní databázový účet – účet je vytvořen,
- neaktivní databázový účet – účet doposud neexistuje,
- žádost se vyřizuje – žádost byla odeslána, čeká se na její schválení,
- účet je zamknut (výjimečný případ) – při opakovaném zadání nesprávného hesla.

Přehled účtů – poskytuje uživateli přehled o aktivních účtech mu náležících. Je zde možnost restartování hesla (funkce 5.4.2/a), dále lze odeslat žádost na změnu údajů či vlastností účtu s tím, že uživatel musí zadat důvod k editaci účtu (funkce 5.4.21/b), a také ho smazat.

Role Databázový a Systémový administrátor

Vytvoření účtu – stránka, která má více účelů, hlavním je jednoduché vytváření databázových účtů (funkce 5.4.5/a). Stejný formulář se používá i pro editaci databázových účtů.

Hromadné vytváření účtů – stránka určená pro hromadné vytváření účtů na základě přiloženého seznamu. V dnešní době se počítá se seznamy vygenerovanými školním systémem Edison [28].

Žádosti – stránka pro správu žádostí jak o vytvoření účtu, tak i změnu údajů databázového účtu, stránka taktéž obsahuje historii schválených či zamítnutých žádostí (funkce 5.4.6).

Synchronizace – stránka sloužící pro synchronizaci účtů, které jsou přebytečné buď na straně databázových systémů, nebo na straně systému DB Manager. Seznam přebytečných účtů, zobrazený na stránce, je vygenerovaný jednou ze služeb komponenty DB Services (viz kapitola 7.2). Přičemž stránka umožňuje uživateli rozhodnout, jestli se má přebytečný účet smazat nebo synchronizovat (vytvořit jej v daném databázovém systému nebo systému DB Manager). Vzhledem k tomu, že databázové systémy obsahují celou škálu systémových účtů, je možné tyto účty označit jako ignorované a systém s nimi již nebude při další synchronizaci počítat.

Databázové účty – jedná z hlavních stránek, poskytující seznam všech spravovaných databázových účtů vytvořených v systému DB Manager. Stránka umožňuje jednotlivé účty mazat, upravovat jejich vlastnosti nebo jim restartovat heslo. Na stránce se nachází komplexní filtr pro co nejrychlejší dohledání hledaného účtu (funkce 5.4.5/b, c, d, e, ,f). Za zmínku také stojí sada hromadných operací, které je možné nad vybranými databázovými účty provádět.

Log – stránka, která poskytuje informace o procesech prováděných v systému DB Manager.

Role Systémový administrátor

Správa serverů – stránka sloužící ke správě databázových systémů a jejich přidávání, editace či mazání. Oznamuje dostupnost jednotlivých systémů a existenci metod pro správu účtů na jednotlivých databázových systémech. V případě jejich neexistence (například z důvodu přeinstalace databázového systému) stránka poskytuje možnost jejich vytvoření. Každý server má určeného správce, kterému se posílá upozornění v případě nedostupnosti jemu přidělenému databázového systému.

Správa uživatelů – slouží k udělování práv jednotlivým uživatelům obsluhujícím systém DB Manager (funkce 5.4.7).

Konfigurace systému – stránka sloužící k úpravám základního nastavení systému DB Manager jedná se o nastavení jako email odesílatele, délka hesla. Mimo jiné se zde nastavuje komponenta DB Services.

7.2 DB Services

Jelikož systém DB Manager je plně závislý na interakci uživatelů, bylo potřeba vymyslet určitý mechanismus, který by byl schopen provádět některé operace automaticky s určitým intervalem. Pro tyto účely vznikl program nazvaný DB Services, který v současné době umožňuje provádět čtyři operace:

- test připojení,
- synchronizaci,
- kontroly platností účtů,
- zálohování.

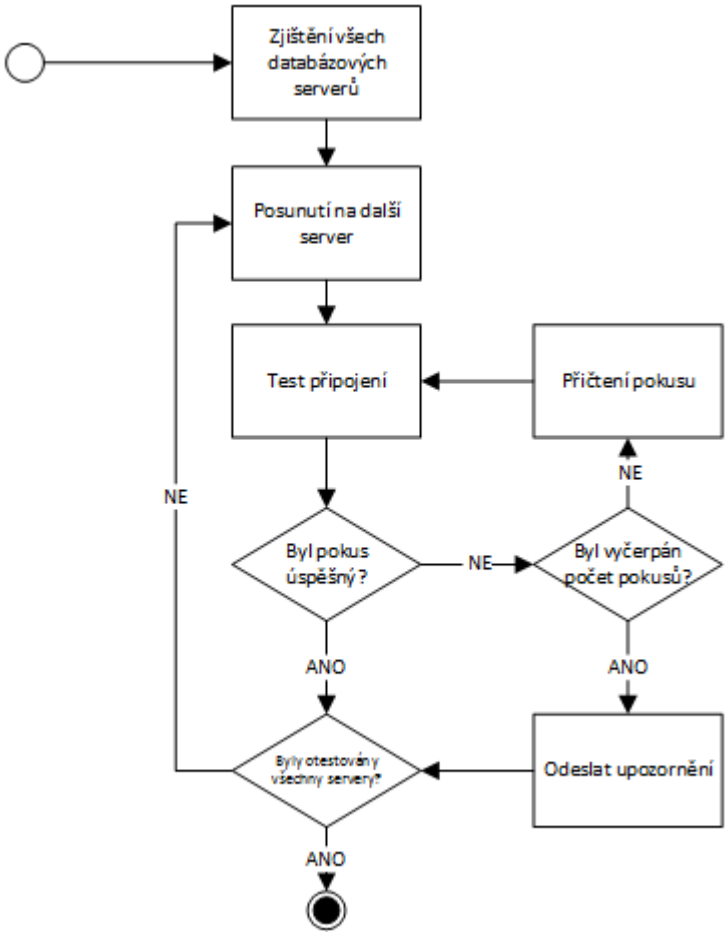
Program je realizován (pro jeho jednoduché zavedení) jako konzolová aplikace. Pro rozlišení operace, kterou má aplikace provádět, je potřeba změnit argument na první pozici pro spuštění aplikace. Jsou přijímány následující argumenty:

- „conn“ – spuštění testu připojení,
- „sync“ – spuštění synchronizace,
- „exp“ – spuštění kontroly platnosti účtů,
- „back“ – spuštění zálohování.

7.2.1 Test připojení

První funkce DB Services je kontrola dostupnosti databázových serverů. Důvodem k zavedení je zamezit dlouhým prodlevám v případě nedostupnosti jednoho ze serverů. Nastávaly totiž situace, kdy nebyl některý server dostupný, avšak dovědět se o tom šlo pouze ručním připojením, nebo na něčí popud. Služba funguje tak, že se zjistí všechny informace o databázových serverech a následně se spouští cyklicky test připojení. Jelikož se za dobu testování zjistilo, že některé databázové systémy jsou občas nedostupné jen krátkodobě (první připojení je neúspěšné, ale druhé za ním je v pořádku), opakuje se neúspěšný pokus tolikrát, kolikrát to systémový administrátor nastaví v sekci Konfigurace

systému. Mezi prováděním těchto pokusů se zavádí i zpoždění pomocí funkce sleep(). Čas zpoždění je taktéž definován v Konfiguraci systému. Pokud ani po těchto pokusech nebude databázový systém dostupný, posílá se automaticky zpráva s upozorněním správci daného databázového systému.



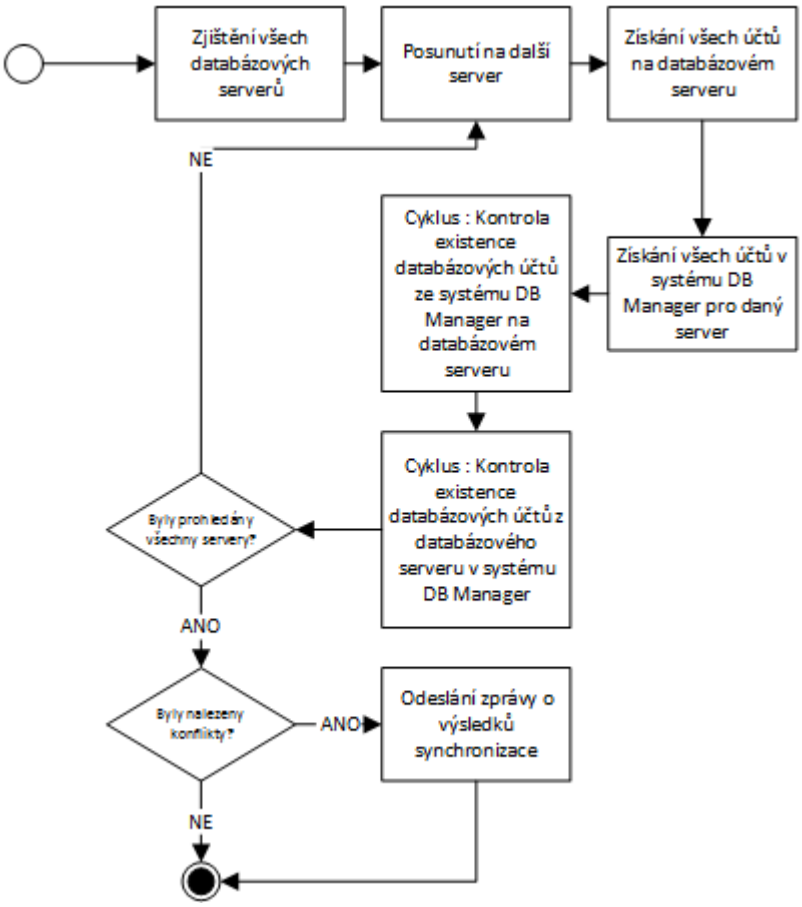
Obrázek 7.3: *Proces testování připojení*

7.2.2 Synchronizace

Další funkcí, kterou je potřeba provádět periodicky je synchronizace. Po ostrém zavedení systému DB Manager vznikly určité zmatky, kdy některé účty byly vytvořeny ručně a některé již v systému DB Manager. To zapříčinilo situaci, kdy systém DB Manager předpokládal, že na databázovém serveru databázový účet neexistuje a nabízel možnost ho vytvořit. Což samozřejmě skončilo chybou, protože daný databázový účet fyzicky na databázovém serveru již existoval. Opakem toho je situace, kdy by někdo smazal účet fyzicky z databázového systému, ale v systému DB Manager by stále zůstával jeho záznam. V tomto případě by nastal problém při mazání daného databázového účtu, protože by se systém snažil smazat účet, který se již fyzicky na databázovém serveru nenachází. Takové účty nazýváme konfliktní.

Proto bylo potřeba vymyslet mechanismus, který na tyto konfliktní účty upozorní a umožní je určitým způsobem zpracovat. Řešení tohoto problému znamená kontrolovat účty, které jsou v systému DB Manager a nejsou v databázovém serveru a také účty, které jsou v databázovém serveru, ale nejsou v systému DB Manager. Pokud se zjistí konflikt, odešle se automaticky zpráva s upozorněním aktuálnímu správci systému.

Je zde ale potřeba vyřešit jeden problém. Každý databázový server má předem definované systémové účty, které se vytvářejí automaticky při generování databáze nebo instance serveru. Nazývejme je systémovými účty. Současné řešení by tyto účty detekovalo jako konfliktní. Proto je vytvářen seznam ignorovaných účtů, které má tato synchronizace ignorovat a neuvádět jako konfliktní. Úlohou tohoto automatu je pouze dané účty porovnat a vytvořit seznam konfliktních účtů. O tom, jak se s těmito účty naloží, rozhoduje administrátor v sekci Synchronizace.

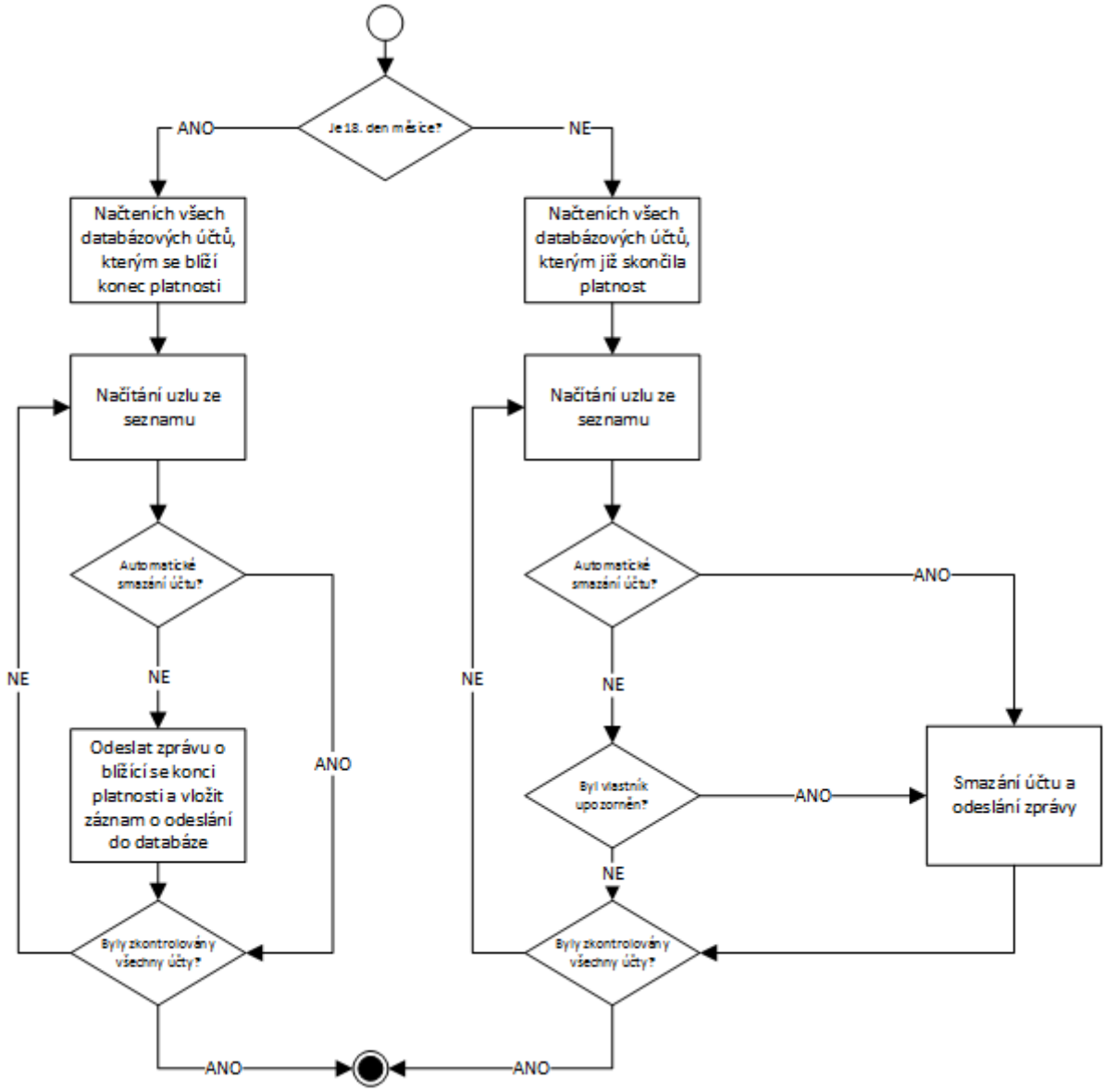


Obrázek 7.4: *Proces provádění synchronizace databázových účtů*

7.2.3 Kontrola platnosti účtů

Při vytváření každého databázového účtu se s ním zavádí datum platnosti a také zda se má účet automaticky po uplynutí této doby smazat. Systém DB Manager nastavuje tyto platnosti vždy na poslední den vybraného měsíce. K tomu byl i zaveden algoritmus, který s těmito platnostmi pracuje a

kontroluje vzhledem k současnému datu. Kontrola má dvě fáze. V první fázi se v 18. den měsíce kontrolují účty, které nemají nastavené automatické smazání a kterým by koncem tohoto měsíce skončila platnost. Pokud je takovýto účet nalezen, posílá se zpráva jeho majiteli o blížícím se konci platnosti s možností jeho prodloužení. Ve druhé fázi se již počítá s tím, že majitel byl upozorněn a účet si buď prodloužil, nebo předpokládá jeho smazání. Pokud je tedy takový databázový účet nalezen a je zde i záznam o tom, že byl uživatel na blížící se konec platnosti upozorněn, databázový účet se smaže a odešle majitel účtu zprávu o jeho smazání. Tyto dvě fáze se opakují každý měsíc a zaručují aktuálnost databázových účtů.



Obrázek 7.5: *Proces kontroly platnosti databázových účtů*

7.2.4 Zálohování

Jako poslední součást programu pro automatické provádění procesů je provádění zálohování. Při každém vytváření databázového učtu se spolu s dalšímu údaji ukládá informace, zdali chce uživatel svůj účet zálohovat nebo ne. Proces funguje tak, že na všech databázových systémech jsou vyhledány databázové účty, které mají nastaveno zálohování a dle získaných informací je provedeno zálohování a odeslána informační zpráva o provedení zálohování.

7.3 Ostatní komponenty

Tato kapitola se zaobírá komponenty, které byly vytvořeny v rámci systému DB Manager a jsou také využívány v dalších systémech umístěných do databázové skupiny. Jejich první verze vznikla jako přímá součást systému DBManager, po jejich osvědčení se rozhodlo o jejich použití i v ostatních systémech databázové skupiny. Já jsem poskytl zdrojové soubory a Ing. Pavel Bednář provedl jejich vylepšení a optimalizaci pro použití i na ostatních systémech.

7.3.1 MailSender

V prvotní verzi systému DBManager měly všechny posílané informační emaily jednoduchou strukturu bez HTML prvků. Důvodem k tomuto omezení byl poštovním systém Horde, který sloužil k přístupu ke školnímu emailu (počítá se s tím, že větší část studentů přistupuje na email přes webové rozhraní). Většina moderních poštovních klientu podporuje implicitně HTML formátování emailu, avšak Horde dokázal zobrazovat emaily nativně pouze bez HTML formátování. Pokud bylo tedy v emailu využito HTML formátování, v Horde se jevil email na první pohled jako prázdný. S nástupem nového poštovního systému Roundcube, který již podporuje HTML formátování implicitně, se nabídla možnost dát posílaným emailům styl a také sjednotit jejich vzhled s jinými systémy například UDBS Analyzer, DB Education či PD Pattern.

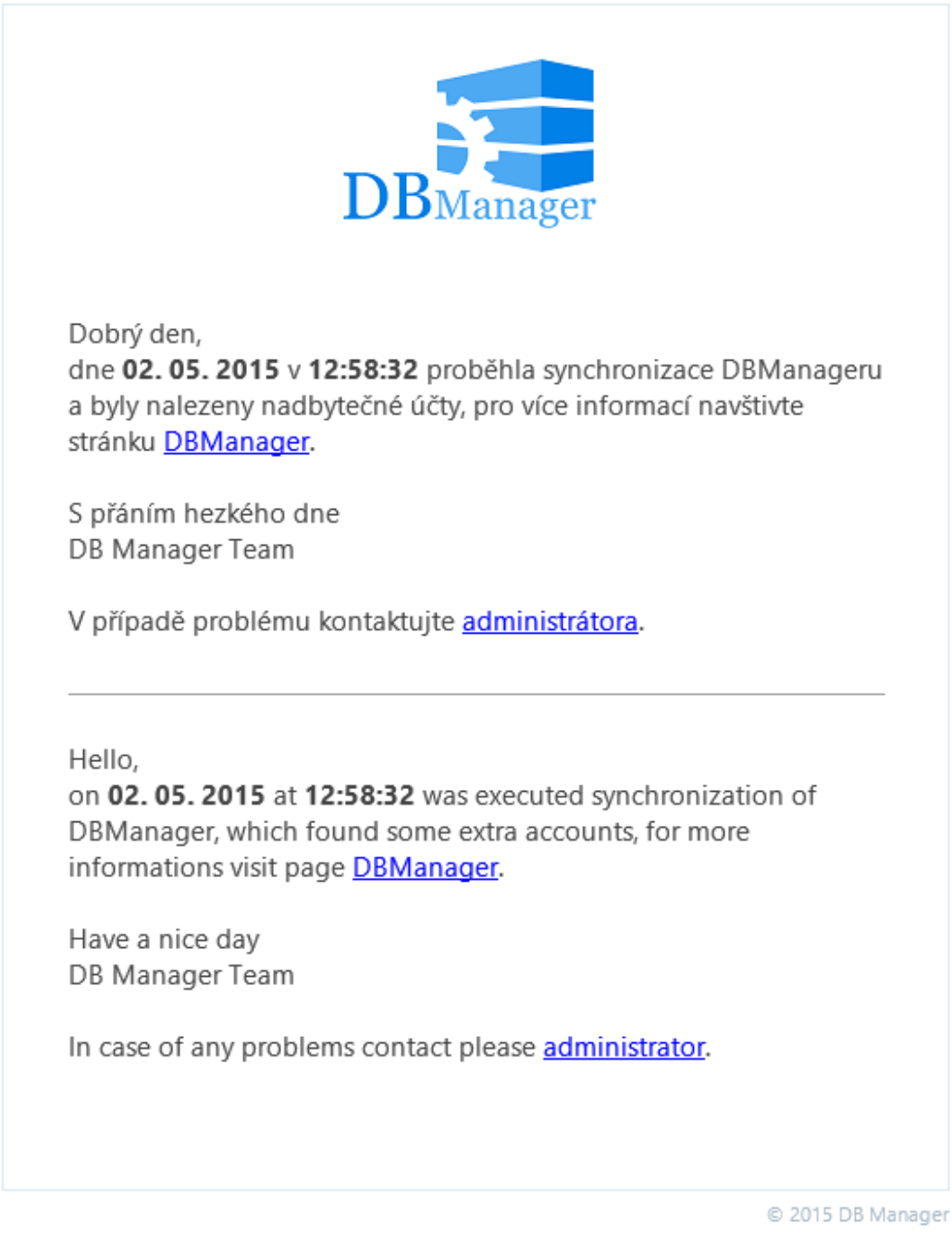
Použití této komponenty je následující. Jako první je potřeba si vytvořit instanci MailSender, určující email odesílatele. Dále je potřeba vytvořit samostatnou zprávu za pomocí DbEduTemplate, která přijímá následující parametry:

- **subject** (string) – název předmětu zprávy,
- **body** (string) – text zprávy v českém jazyce,
- **body_en** (string) – text zprávy v anglickém jazyce,
- **application** (eApplication) – typ aplikace, ze které se zpráva odesílá (jiné logo v hlavičce emailu).

eApplication je definována třemi typy (každý představuje jiný Dbedu systém):

- eApplication.Dbedu
- eApplication.Dbman
- eApplication.UdbsAnalyzer

Vytváření zprávy funguje tak, že vstupní prvky subject, body a body_en jsou zabaleny do HTML šablony a následně vráceny jako čistý HTML text. Dále se vytvoří instance MailMessage, představující samotnou zprávu vytvořenou v předešlém kroku a také email příjemce. Pak stačí pouze zavolat MailSender s parametrem MailMessage a zprávu odeslat. Tato funkce má návratový typ bool, čili ohlašuje, zda bylo odeslání úspěšné nebo ne. Na obrázku 7.6 je možno vidět vhled emailu posílaného systémem DB Manager.



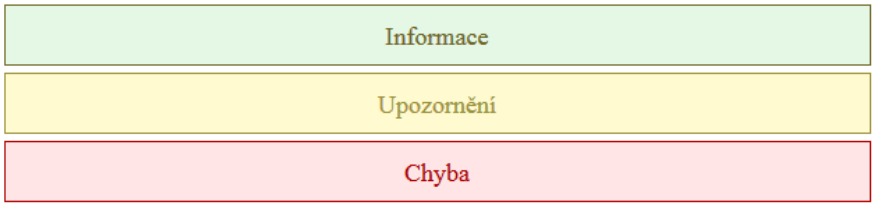
Obrázek 7.6: *Vzhled emailu posílaného systémem DB Manager*

7.3.2 InfoBox

Další komponentou využívanou v systémech databázové skupiny je InfoBox. Jak již z názvů vypovídá, jedná o prostředek sloužící k zobrazování informačních oken uživateli. InfoBox je dán třemi typy zobrazení:

- **Info** – upozorňuje uživatele o úspěšném provedení operace
- **Warning** – upozorňuje uživatele na nezávažnou chybu
- **Error** – upozorňuje uživatele na závažnou chybu

Dále může být InfoBox různě přizpůsobován, může obsahovat tlačítka (Ok, Ano, Ne), může mu být nastaveno automatické skrytí po uplynutí určitého času. Komponenta je realizována jako ASP.NET User Control a pro její zavedení stačí registrovat komponentu v hlavičce ASP.NET stránky a vložit ji na požadovanou pozici na stránce. Infobox může informovat i o probíhajícím procesu (viz obrázek 7.8).



Obrázek 7.7: *Infobox – ukázka všech tří typů upozornění*

Obrázek 7.8: *Infobox - synchronizace*

7.3.3 VsbLogin

Jedná se o bezpečnostní prvek systému DBManager. VsbLogin má za úkol obsluhovat veškeré operace prováděné nad univerzitním LDAP systémem. Jedná se hlavně o kontrolu přihlášení a také o získávání informaci o uživateli z LDAP serveru.

Přihlašovací systém

Přihlašovací systém se skládá z více částí, které dohromady tvoří funkční celek. Jeho hlavní funkcí je samotné ověření přihlášení uživatele. To je prováděno za pomocí dotazování se nad LDAP serverem. Jako první se provádí kontrola existence uživatelského jména v LDAP adresáři. Pokud je uživatel nalezen, postupuje se k ověření správnosti hesla. Zde se již ale nevyužívá samotného názvu uživatele, ale vytváří se jednoznačné rozlišovací jméno uživatele (Distinguished name). Pokud je ověření provedeno úspěšně, je přihlášení považováno za platné. To ovšem ještě není vše, všechny systémy databázové skupiny jsou nastaveny tak, aby používaly stejné autorizační cookie, čili pokud je uživatel přihlášen na webu dbedu.cs.vsb.cz, je jeho přihlášení registrováno i v systému DBManager. Po platném přihlášení je nastavováno autorizační cookie pro daného uživatele a to dvěma způsoby:

1. uživatel zaškrtne možnost trvalého přihlášení a je mu nastavena doba přihlášení na 2 roky,
2. uživatel nezaškrtne možnost trvalého přihlášení a je mu nastavena doba přihlášení na 2 hodiny.

Pokud je vše nastaveno a ověřeno, přichází na řadu určení uživatelovy role. Jak jsem již popsal v předešlé kapitole, systém DBManager rozlišuje tři role. Při každém spuštění aplikace je volána funkce pro zjištění všech rolí v systému a ty jsou následně nastaveny pro aktuální instanci webu. Implementaci si lze prohlédnout v příloze M.

Informace o uživateli

Jak již bylo několikrát psáno, systém DBManager pracuje s loginy uživatelů ve formátu ABC0001. To ovšem nevypovídá nic o uživateli a jeho údajích. Pro získání těchto informací je potřeba procházet adresář LDAP, který všechny tyto informace obsahuje. K těmto účelům slouží třída UserInfo komponenty VsbLogin, která přijímá jako parametr login uživatele. Třída se podobně jako při přihlašovaní snaží nalézt záznam o uživateli. Pokud jej nalezne, je potřeba dostat z tohoto záznamu informace o uživateli v rozumném formátu. Konkrétně se nám jedná o tyto údaje:

- **cn** – login uživatele,
- **givenname** – jméno uživatele,
- **sn** – příjmení uživatele,
- **mail** – email uživatele.

Zdrojový kód pro zpracování záznamu vypadá následovně:

```
SearchResult result = search.FindOne();
this.Login = result.Properties["cn"][0].ToString();
this.FName = result.Properties["givenname"][0].ToString();
this.LName = result.Properties["sn"][0].ToString();
```

```
this.Email = result.Properties["mail"][0].ToString();
```

Jelikož každý záznam může obsahovat více emailů, nebo více fakult, je vyhledání zaměřeno pouze na ten, který je na první pozici. Výsledkem je tedy objekt UserInfo obsahující získané údaje o uživateli. Pokud by informace nebyly k nalezení, tedy zadaný uživatel v LDAP adresáři neexistuje, bude objekt null. Tato funkce je využívána v systému pokaždé, kdy potřebujeme zjistit informace o uživateli (jeho email, jméno, příjmení atd.).

7.4 XML Action

Po prvním zavedení systému DB Manager do zkušebního provozu byly zjištěny různé chyby. Některé chyby byly zapříčiněné nedostatky DB Manageru, ale některé se týkaly samotných databázových systémů. Vyskytovaly se chyby jako nedostupnost databázového systému, nebo neúspěch při vykonávaní procedur. Pro tyto a jiné účely vznikla komponenta nazvaná XML Action, která má úkol ukládat všechny vykonané operace a vyvolané chyby v systému DB Manager. Informace o jednotlivých činnostech se ukládají jak z důvodu dohledání *„kdo“* provedl *„který“* proces, tak pro budoucí vytváření statistik. Chyby jsou zachyceny v blocích try-catch a nebo jako chyba ASP.NET v metodě Application_Error v Global.asax, kdy při vyvolání této metody je uživatel přesměrován na informační stránku, informující o výskytu chyby (ne však její znění). V konečném důsledku se všechny tyto záznamy zobrazují na stránce Log s možností filtrace dle typu.

XML Action záznam obsahuje 4 atributy:

- **login** – uživatel, který operaci, případně chybu vyvolal
- **proces** – název procesu, případně chyby
- **message** – zkrácené znění zprávy,
- **fullMessage** – celé znění zprávy,
- **type** – typ záznamu.

Typy záznamů:

- **INFO** – informační záznam provedené operace, například vytvoření uživatele,
- **ASP_ERROR** – chyba zachycená v metodě Application_Error,
- **SERVER_ERROR** – chyby v blocích try-catch,
- **CON_TEST** – testování dostupnosti serverů.

Zachycení chyby:

```
new XML_Action().InsertAction(DateTime.Now, this.GetType().Name +
"_" + MethodBase.GetCurrentMethod().Name, e.ToString(),
"SERVER_ERROR");
```

Popis jednotlivých metod:

- **this.GetType().name** – získání názvu třídy, odkud byla výjimka vyvolána,
- **MethodBase.GetCurrentMethod().Name** – získání názvu metody či funkce, ve které byla výjimka vyvolána,
- **e.ToString()** – znění chyby, může mít i jiný formát, než je v příkladu.

Testování uzavírání připojení – CON_TEST

Po nasazení systému DB Manageru jsme se z počátku potýkali s problémy u připojení k databázovým serverům Oracle. Z výpisu chyb se jako problém jevila skutečnost, že v určitých případech údajně nedocházelo k ukončování připojení a proto systémy Oracle nedovolovaly vytvoření dalšího připojení z důvodu vyčerpání limitu připojení. Pro ověření této teorie bylo využito XML_Action s typem CON_TEST, představující záznam o připojení a ukončení spojení. V praxi to vypadalo tak, že pro každé vytvoření připojení bylo vyhledáváno i ukončení připojení. Pokud by ukončení nebylo nalezeno, tak právě ta metoda, u které by byl evidován tento záznam, by způsobovala ony výpadky při připojování k databázovým systémům Oracle (způsobovala by vyčerpání prostředků pro připojení). Tímto způsobem byly odladěny všechny chyby související s tímto problémem.

8 Fond rozvoje vysokých škol

Systém DB Manager byl a zároveň ještě je součástí projektu pro Fond Rozvoje Vysokých Škol financovaným Ministerstvem školství, mládeže a tělovýchovy. Celý projekt byl vytvořen za účelem podpory výuky databázových předmětů na katedře informatiky Vysoké školy báňské a jeho hlavním řešitelem byl Ing. Pavel Bednář. Jednalo se o vývoj jak systému DB Manager, tak i rozšíření webového portálu dbedu.cs.vsb.cz a vývoj aplikace pro hodnocení projektů, zadání a ukázkové aplikace pro předmět Úvod do databázových systémů.

Prvním rokem (2013/2014) procházel DB Manager celkovou optimalizací a řešením chyb, na které se přišlo při prvním nasazení. Byla doplněna podpora synchronizace, přidány další databázové systémy a vytvořena online nápověda. V letech 2013/2014 byl projekt úspěšně obhájen. Druhý rok (2014/2015) je zaměřen na vývoj již výše zmiňované podaplikace DB Services, která má za úkol automatizaci některých procesů jako kontrola dostupnosti systémů, apod.

9 Závěr

Cílem této práce bylo vymyslet mechanismus umožňující správu databázových účtů a databází za pomocí určitého uživatelského rozhraní. Toho bylo dosaženo dodržením bodů definovaných v úvodu práce.

Nastudování základních informací o databázových systémech stanovilo zaměření se na klient-server databázové systémy. Tento typ systému umožňuje vzdálený přístup, který přesně vyhovuje pro účely výuky databázových a informačních předmětů. Na mysli je umožnění vzdáleného přístupu k těmto databázovým systémům studentům či zaměstnancům.

Kapitola obsahující informace o správě databázových systému umožnila začít přemýšlet nad tím, jak by mohla být samotná realizace správy vyvíjena. Za pomocí určení si základních pojmů, přesně definujících jednotlivé prvky administrace databázových systémů, vznikl náhled na to, v jaké posloupnosti musí být prováděny příkazy. Tak aby bylo možno vytvářet databázové účty s přesně definovanými parametry a právy. Pro tyto účely vzniklo rozhraní IConnections určující, které metody a funkce musí dané třídy (zastupující jednotlivé databázové systémy) implementovat, aby byla zajištěna funkčnost celé komponenty.

Při analýze informací o konkrétních databázových systémech a jejich administraci bylo zjištěno několik faktů. Prvním je, že všechny poskytují velké množství ovladačů, umožňujících připojit se z velké škály platforem a programovacích jazyků. Všechny tyto ovladače navíc sdílejí stejnou strukturu, proto je z tohoto hlediska jejich zavedení velice jednoduché a není potřeba studovat jejich funkčnost. Druhým faktem je, že každý databázový systém má jinak řešenou administrační stránku a právě toto dělalo tuto kapitolu nejvíce náročnou. Přístup k databázím je realizován více způsoby, někdy se uživateli vytváří jeho vlastní databáze, jindy se uděluje pouze přístup k tabulkovému prostoru z důvodu časové náročnosti při vytváření samotné databáze.

Pro obsluhu komponenty pro správu databázových systému vznikly dvě rozhraní DB Manager a DB Services. Systém DB Manager umožňuje správu databázových účtů z webového prostředí, poskytuje všechny funkce nutné pro správu a pracuje s několika rolemi přihlášení. DB Services vzniklo za účelem automatizace procesů. Umožňuje za pomocí plánovače provádět periodicky určité operace a není zde potřeba interakce uživatele. Jedná se o operace testování připojení k databázovým systémům, kontrola platnosti databázových účtů, synchronizace a zálohování.

Tento projekt se také stal součástí většího projektu pro podporu výuky databázových předmětů na katedře informatiky financovaným Fondem rozvoje vysokých škol Ministerstvem školství, mládeže a tělovýchovy. Projekt měl přesně dané body, které definovaly jak opravy chyb, tak i vývoj a zavádění nových prvků do systému DB Manager či DB Services. Tyto body bylo nutné splnit v přesně daných časových intervalech. Projekt byl obhájen před komisí FRVŠ a také byl pro rok 2014/2015 znovu podporován.

Tato práce pro mne byla obrovským přínosem jak z hlediska načerpání nových znalostí z oblasti databázových systémů, tak i možnost práce v týmu. Možnost prezentovat tuto práci na dni otevřených dveří Fakulty elektrotechniky a informatiky mne velice naplnila a je pro mne potěšující, že

systém je hojně využíván jak zaměstnanci, tak i studenty. Ti jej využívají čím dál víc pro vytváření databázových účtů potřebných i pro jiné předměty, které nebyly zmíněny v úvodu.

Použitá literatura

[1] CHONG, Raul F, Ian HAKES a Rav S AHUJA. *Začínáme s DB2 Express-C: kniha od komunity pro komunitu*. 3. vyd. Praha: DNS, 2009, 287 s. ISBN 978-802-6018-483.

[2] LIU, Raul Chong and Clara. *DB2 essentials: understanding DB2 in a big data world*. 3rd edition. Upper Saddle River, NJ: IBM Press, Pearson, 2013. ISBN 01-334-6190-4.

[3] ATKINSON, Paul. *Beginning Microsoft SQL Server 2012 Programming*. Indianapolis: Wiley, 2012, xxx, 833 s. ISBN 978-1-118-10228-2.

[4] BEN-GAN, Itzik. *Microsoft SQL Server 2012 high-performance T-SQL using window functions*. Sebastopol, Calif.: O'Reilly Media, 2012, xvii, 221 p. ISBN 07-356-5836-6.

[5] SCHWARTZ, Baron, Peter ZAITSEV a Vadim TKACHENKO. *High performance MySQL*. 3rd ed. Cambridge [Mass.]: O'Reilly, 2012, xxviii, 793 p. ISBN 14-493-1428-7.

[6] DUBOIS, Paul. *MySQL Cookbook*. 1st ed. Sebastopol, CA: O'Reilly, 2003, xxvii, 992 p. ISBN 05-960-0145-2.

[7] LONEY, Kevin. *Oracle database 11g the complete reference*. New York: McGraw-Hill, 2009. ISBN 978-007-1598-767.

[8] GREENWALD, Rick. *Professional Oracle programming*. Indianapolis, Ind.: Wiley Pub., 2005, xxxii, 758 p. ISBN 07-645-7482-5.

[9] MATTHEW, Neil a Richard STONES. *Beginning databases with PostgreSQL: from novice to professional*. 2nd ed. Berkeley, CA: Apress, 2005, xxiv, 637 p. ISBN 15-905-9478-9.

[10] SMITH, Gregory. *PostgreSQL 9.0 high performance: accelerate your PostgreSQL system and avoid the common pitfalls that can slow it down*. 1st pub. Birmingham: Packt Publishing, 2010, xiii, 442 s. ISBN 978-184-9510-301.

[11] OWENS, Michael. *The definitive guide to SQLite*. New ed. Berkeley, Calif: Apress, 2006. ISBN 978-159-0596-739.

[12] KREIBICH, Jay A. *Using SQLite*. 1st ed. Beijing: O'Reilly, 2010, xx, 503 s. ISBN 978-059-6521-189

[13] ŠARMANOVÁ, Jana. *Databázové a informační systémy*. Ostrava: Vysoká škola báňská - Technická univerzita, 2007, 1 CD-R. ISBN 978-80-248-1499-5.

Internetové zdroje

[14] Oracle | Hardware and Software, Engineered to Work Together. [online]. [cit. 2015-04-22]. Dostupné z: http://www.oracle.com

[15] IBM Knowledge Center. [online]. [cit. 2015-04-22]. Dostupné z: http://www-01.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.kc.doc/welcome.html

[16] Co je systém SQL Server 2012–2014 | Microsoft. [online]. [cit. 2015-04-22]. Dostupné z: http://www.microsoft.com/cs-cz/server-cloud/products/sql-server/

[17] Jak na MS SQL – uživatelé a práva | Interval.cz. [online]. [cit. 2015-04-22]. Dostupné z: https://www.interval.cz/clanky/jak-na-ms-sql-uzivatele-a-prava/

[18] MySQL :: The world's most popular open source database. [online]. [cit. 2015-04-22]. Dostupné z: http://www.mysql.com/

[19] PostgreSQL: The world's most advanced open source database. [online]. [cit. 2015-04-22]. Dostupné z: http://www.postgresql.org/

[20] Systém řízení báze dat – Wikisofia. [online]. [cit. 2015-04-29]. Dostupné z: https://wikisofia.cz/index.php/Syst%C3%A9m_%C5%99%C3%ADzen%C3%AD_b%C3%A1ze_dat

[21] M. Kratky, R. Baca: Databazove systemy. [online]. [cit. 2015-05-01]. Dostupné z: http://dbedu.cs.vsb.cz/SubPages/OpenFile.aspx?file=book/dbcb.pdf

[22] *Firebird: The true open source database for Windows, Linux, Mac OS X and more* [online]. [cit. 2015-05-03]. Dostupné z: http://www.firebirdsql.org/

[23] *Home page | Sybase* [online]. [cit. 2015-05-03]. Dostupné z: http://www.sybaseproducts.com/cs/node/1

[24] *SQLite Home Page* [online]. [cit. 2015-05-03]. Dostupné z: https://www.sqlite.org/

[25] *Databázový software a aplikace | Microsoft Access* [online]. [cit. 2015-05-03]. Dostupné z: https://products.office.com/cs-cz/access

[26] *Oracle Berkeley DB* [online]. [cit. 2015-05-03]. Dostupné z: http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html

[27] *MySQL :: MySQL as an Embedded Database* [online]. [cit. 2015-05-03]. Dostupné z: https://www.mysql.com/oem/

[28] *Informační systém pro evidenci studia a výuky* [online]. [cit. 2015-05-05]. Dostupné z: http://edison.vsb.cz/

Seznam příloh

Součástí diplomové práce je CD/DVD.

Příloha A: Microsoft SQL Server – procedura pro vytvoření uživatelského účtu

```
CREATE PROCEDURE [dbo].[CreateUser] @p_UserName
VARCHAR(20),@p_passwd VARCHAR(20), @p_Quota INT

AS

DECLARE @dir NVARCHAR(4000)


EXECUTE master.dbo.xp_instance_regread N'HKEY_LOCAL_MACHINE',
N'Software\Microsoft\MSSQLServer\Setup', N'DefaultData', @dir
output;


IF (@dir IS NULL)

BEGIN

     exec master.dbo.xp_instance_regread N'HKEY_LOCAL_MACHINE',
N'Software\Microsoft\MSSQLServer\Setup', N'SQLPath', @dir output;

     SET @dir = @dir + '\DATA\';

END

ELSE

BEGIN

     SET @dir = @dir + '\';

END;


PRINT 'PATH: ' + @dir;


DECLARE @cmd NVARCHAR(1000)


SET @cmd = 'CREATE DATABASE ' + @p_UserName + ' ON PRIMARY
(NAME=N''' + @p_UserName + ''', FILENAME=N''' + + @dir + @p_UserName
+ '.mdf'', MAXSIZE = ' + CAST(@p_Quota AS varchar) + 'MB);';

PRINT 'COMMAND: ' + @cmd;

EXEC sp_executesql @cmd;


SET @cmd = 'CREATE LOGIN ' + @p_UserName + ' WITH PASSWORD = ''' +
@p_passwd + ''', CHECK_POLICY = OFF;';

EXEC sp_executesql @cmd;
```

```
SET @cmd = 'ALTER LOGIN ' + @p_UserName + ' WITH DEFAULT_DATABASE =
' + @p_UserName + ';';
EXEC sp_executesql @cmd;

SET @cmd = 'USE ' + @p_UserName + ';' + 'CREATE USER ' + @p_UserName
+ ' FOR LOGIN ' + @p_UserName + ' WITH DEFAULT_SCHEMA = dbo;'; -- +
@p_UserName + ';';
EXEC sp_executesql @cmd;

SET @cmd =  'USE ' + @p_UserName + ';' + 'GRANT CREATE TABLE To ' +
@p_UserName + ';';
EXEC sp_executesql @cmd
SET @cmd = 'USE ' + @p_UserName + ';' + 'GRANT ALTER To ' +
@p_UserName + ';';
EXEC sp_executesql @cmd
SET @cmd = 'USE ' + @p_UserName + ';' + 'GRANT CONTROL To ' +
@p_UserName + ';';
EXEC sp_executesql @cmd

-- it is necessary for bulkload
SET @cmd = 'USE ' + @p_UserName;
EXEC sp_executesql @cmd;
EXEC sp_addsrvrolemember @p_UserName, 'bulkadmin';
```

Příloha B: Microsoft SQL Server – procedura pro smazání uživatelského účtu

```
CREATE PROCEDURE [dbo].[DropUser] @p_UserName VARCHAR(20)
AS
DECLARE @cmd NVARCHAR(200);
DECLARE dropCurs CURSOR FOR SELECT session_id FROM
sys.dm_exec_sessions WHERE login_name = @p_UserName;
DECLARE @id INT;

OPEN dropCurs;
FETCH NEXT FROM dropCurs INTO @id;
WHILE @@FETCH_STATUS = 0
```

```
    BEGIN
      SET @cmd = 'KILL ' + CAST(@id AS NVARCHAR(10)) + ';';
      EXEC sp_executesql @cmd;
    FETCH NEXT FROM dropCurs INTO @id;
END
CLOSE dropCurs;
DEALLOCATE dropCurs;

SET @cmd = 'USE ' + @p_UserName + ';DROP USER ' + @p_UserName + ';';
EXEC sp_executesql @cmd;

SET @cmd = 'DROP LOGIN ' + @p_UserName + ';';
EXEC sp_executesql @cmd;

SET @cmd = 'DROP DATABASE ' + @p_UserName + ';';
EXEC sp_executesql @cmd;
```

Příloha C: Microsoft SQL Server – procedura pro změnu kvóty

```
CREATE PROCEDURE SetQuota @p_UserName VARCHAR(20), @p_Quota INT
AS
DECLARE @cmd NVARCHAR(1000)
SET @cmd = ' ALTER DATABASE ' + @p_UserName + ' MODIFY FILE (NAME =
''' + @p_UserName + ''', MAXSIZE=' + CAST(@p_quota AS varchar) +
'MB);'
EXEC sp_executesql @cmd;
```

Příloha D: Oracle – procedura pro vytvoření uživatelského účtu

```
create or replace PROCEDURE CreateUser
(
username IN VARCHAR2,
pass IN VARCHAR2,
quota IN INT
)
AS
BEGIN
DBMS_OUTPUT.PUT_LINE('CREATE USER ' || username || ' IDENTIFIED BY '
|| pass || ' DEFAULT TABLESPACE USERS TEMPORARY TABLESPACE TEMP
QUOTA ' || quota || 'M ON USERS');

EXECUTE IMMEDIATE('CREATE USER ' || username || ' IDENTIFIED BY ' ||
pass || ' DEFAULT TABLESPACE USERS TEMPORARY TABLESPACE TEMP QUOTA '
|| quota || 'M ON USERS');
EXECUTE IMMEDIATE('GRANT CONNECT TO ' || username);
EXECUTE IMMEDIATE('GRANT RESOURCE TO ' || username);
EXECUTE IMMEDIATE('GRANT CREATE VIEW TO ' || username);
EXECUTE IMMEDIATE('GRANT CREATE TABLE TO ' || username);
EXECUTE IMMEDIATE('GRANT SELECT_CATALOG_ROLE TO ' || username);
EXECUTE IMMEDIATE('GRANT SELECT ANY DICTIONARY TO ' || username);
END;
```

Příloha E: Oracle – procedura pro smazání uživatelského účtu

```
create or replace PROCEDURE DropUser
(
p_username IN VARCHAR2
)
AS
  CURSOR killCurs IS select sid,serial# from v$session where
username = UPPER(p_username);
  p_sid INT;
  p_serial INT;
BEGIN
  FOR killID IN killCurs LOOP
    DBMS_OUTPUT.PUT_LINE('ALTER SYSTEM KILL SESSION ''' ||
killID.sid || ',' || killID.serial# || ''' IMMEDIATE');
    EXECUTE IMMEDIATE ('ALTER SYSTEM KILL SESSION ''' || killID.sid
|| ',' || killID.serial# || ''' IMMEDIATE');
  END LOOP;
  EXECUTE IMMEDIATE ('DROP USER ' || p_username || ' CASCADE');
END;
```

Příloha F: Oracle – procedura pro změnu kvóty

```
create or replace PROCEDURE SetQuota
(
username IN VARCHAR2,
quota IN INT
)
AS
BEGIN
DBMS_OUTPUT.PUT_LINE('ALTER USER ' || username || ' DEFAULT
TABLESPACE USERS TEMPORARY TABLESPACE TEMP QUOTA ' || quota || 'M ON
USERS');
EXECUTE IMMEDIATE('ALTER USER ' || username || ' DEFAULT TABLESPACE
USERS TEMPORARY TABLESPACE TEMP QUOTA ' || quota || 'M ON USERS');
END;
```

Příloha G: PostgreSQL – procedura pro vytvoření uživatelské účtu

```
CREATE EXTENSION IF NOT EXISTS dblink;


CREATE OR REPLACE FUNCTION createuser(username character varying,
pass character varying)
  RETURNS void AS
$BODY$
DECLARE
connection VARCHAR;
BEGIN
connection :='dbname=' || current_database() || ' port=' ||
inet_server_port() ||' user=' || current_user;


PERFORM dblink_exec(connection,'CREATE USER ' || username || ' WITH
password ''' || pass || '''');
PERFORM dblink_exec(connection,'CREATE DATABASE ' || username);


PERFORM dblink_exec(connection,'GRANT CONNECT ON DATABASE ' ||
username || ' TO ' || username);
PERFORM dblink_exec(connection,'GRANT CREATE ON DATABASE ' ||
username || ' TO ' || username);


END;
$BODY$
LANGUAGE plpgsql VOLATILE
```

Příloha H: PostgreSQL – procedura pro smazání uživatelského účtu

```
CREATE EXTENSION IF NOT EXISTS dblink;

CREATE OR REPLACE FUNCTION dropuser(username character varying)
  RETURNS void AS
$BODY$
DECLARE
connection VARCHAR;
BEGIN
connection :='dbname=' || current_database() || ' port=' ||
inet_server_port() ||' user=' || current_user;

PERFORM dblink_exec(connection, 'DROP DATABASE ' || username);
PERFORM dblink_exec(connection, 'DROP USER ' || username);
END;
$BODY$

LANGUAGE plpgsql VOLATILE
```

Příloha I: MySQL – procedura pro vytvoření uživatelského účtu

```
DROP PROCEDURE IF EXISTS CreateUser;


CREATE PROCEDURE CreateUser(user char(10),pass char(20))
BEGIN
                SET @s = CONCAT('CREATE DATABASE ',user);
                PREPARE stmt FROM @s;
                EXECUTE stmt;


                SET @s = CONCAT('GRANT USAGE ON *.* TO ', user, '@',
'''%''',' IDENTIFIED BY ''', pass, '''');
                PREPARE stmt FROM @s;
                EXECUTE stmt;


                SET @s = CONCAT('GRANT SELECT, INSERT, UPDATE,
DELETE, CREATE, ALTER, INDEX, DROP, CREATE TEMPORARY TABLES, SHOW
VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE, CREATE VIEW, EVENT,
TRIGGER ON ',user,'.* TO ', user, '@', '''%''');
                PREPARE stmt FROM @s;
                EXECUTE stmt;
END
```

Příloha J: MySQL – procedura pro smazání uživatelského účtu

```
DROP PROCEDURE IF EXISTS DropUser;

CREATE PROCEDURE DropUser(user char(10))
BEGIN
                SET @s = CONCAT('DROP USER ',user);
                PREPARE stmt FROM @s;
                EXECUTE stmt;

                SET @s = CONCAT('DROP DATABASE ',user);
                PREPARE stmt FROM @s;
                EXECUTE stmt;
END
```

Příloha K: IBM DB2 – procedura pro vytvoření uživatelského účtu

```
--#SET TERMINATOR @
CREATE OR REPLACE PROCEDURE CreateUser(IN username VARCHAR(50), IN
databasename VARCHAR(50))
     LANGUAGE SQL
BEGIN
     EXECUTE IMMEDIATE('GRANT USE OF TABLESPACE USERSPACE1 TO ' ||
username || '');

     EXECUTE IMMEDIATE('CREATE SCHEMA ' || databasename || '
AUTHORIZATION ' || username || '');

     EXECUTE IMMEDIATE('GRANT CONNECT ON DATABASE TO ' || username
|| '');

     EXECUTE IMMEDIATE('GRANT CREATETAB ON DATABASE TO ' || username
|| '');
END
@
```

Příloha L: IBM DB2 – procedura pro smazání uživatelského účtu

```
--#SET TERMINATOR @

CREATE OR REPLACE PROCEDURE CreateUser(IN username VARCHAR(50), IN
databasename VARCHAR(50))

     LANGUAGE SQL

BEGIN

     DECLARE counter INT;


     EXECUTE IMMEDIATE('GRANT USE OF TABLESPACE USERSPACE1 TO ' ||
username || '');


     EXECUTE IMMEDIATE('CREATE SCHEMA ' || databasename || '
AUTHORIZATION ' || username || '');


     EXECUTE IMMEDIATE('GRANT CONNECT ON DATABASE TO ' || username
|| '');


     EXECUTE IMMEDIATE('GRANT CREATETAB ON DATABASE TO ' || username
|| '');

END

@
```

Příloha M: Implementace přihlašovacího systému

```
public static bool Authentificate(string login, string password,
HttpResponse Response,HttpRequest Request, bool persistent)
{
            login = login.ToLower();
            if (Debug || (password != null && password.Trim().Length
> 0 &&  GetInstance().LDAPLogin(login, password)))
            {
                VsbUser lu = new VsbUser(login,
Settings.rolesNamespace);
                string encryptedTicket =
SecurityManager.Encrypt(lu.Serialize(), Settings.EncryptionSeed);
                HttpCookie authCookie =
Response.Cookies.Get(Settings.CookieName);
                authCookie.Value = encryptedTicket;
                if (persistent)
                    authCookie.Expires = DateTime.Now.AddYears(2);
                else
                    authCookie.Expires =
DateTime.Now.Add(Settings.CookieExpiration);
                Response.Cookies.Set(authCookie);
                Request.Cookies.Set(authCookie);
                return true;
            }
            else
            {
                return false;
            }
        }
private bool LDAPLogin(string login, string password)
        {
            LdapConnection con = new LdapConnection(new
LdapDirectoryIdentifier("ldap.vsb.cz", 636));
            con.SessionOptions.SecureSocketLayer = true;
```

```
            con.SessionOptions.VerifyServerCertificate = new
VerifyServerCertificateCallback(ServerCallback);
            con.AuthType = AuthType.Anonymous;

            try
            {
                using (con)
                {
                    con.Bind();
                    try
                    {
                        SearchRequest request = new
SearchRequest("", String.Format("(&(objectClass=Person)(uid={0}))",
login), SearchScope.Subtree);
                        SearchResponse response =
(SearchResponse)con.SendRequest(request);

                        if (response.Entries.Count == 0)
                        {
                            return false;
                        }
                        else
                        {
                            SearchResultEntry entry =
response.Entries[0];
                            string dn = entry.DistinguishedName;

                            con.Credential = new
NetworkCredential(dn, password);
                            con.AuthType = AuthType.Basic;
                            con.Bind();

                            return true;
                        }
                    }
```

```
                catch (DirectoryOperationException)
                {

                }
                catch (LdapException)
                {

                }
                catch (Exception ex)
                {

                }
                return false;

            }
        }
        catch (System.Exception ex)
        {
            return false;

        }
    }
```